US 20140082295A1

(19) **United States**

(12) **Patent Application Publication**

Beard et al.

(10) **Pub. No.: US 2014/0082295 A1**

(43) **Pub. Date: Mar. 20, 2014**

(54) **DETECTION OF OUT-OF-BAND ACCESS TO A CACHED FILE SYSTEM**

(71) Applicant: **NetApp, Inc.**, Sunnyvale, CA (US)

(72) Inventors: **Derek Beard**, Austin, TX (US); **Ghassan Yammine**, Leander, TX (US); **Greg Dahl**, Austin, TX (US)

(73) Assignee: **NetApp, Inc.**, Sunnyvale, CA (US)

(21) Appl. No.: **14/031,023**

(22) Filed: **Sep. 18, 2013**

Related U.S. Application Data

(60) Provisional application No. 61/702,692, filed on Sep. 18, 2012.

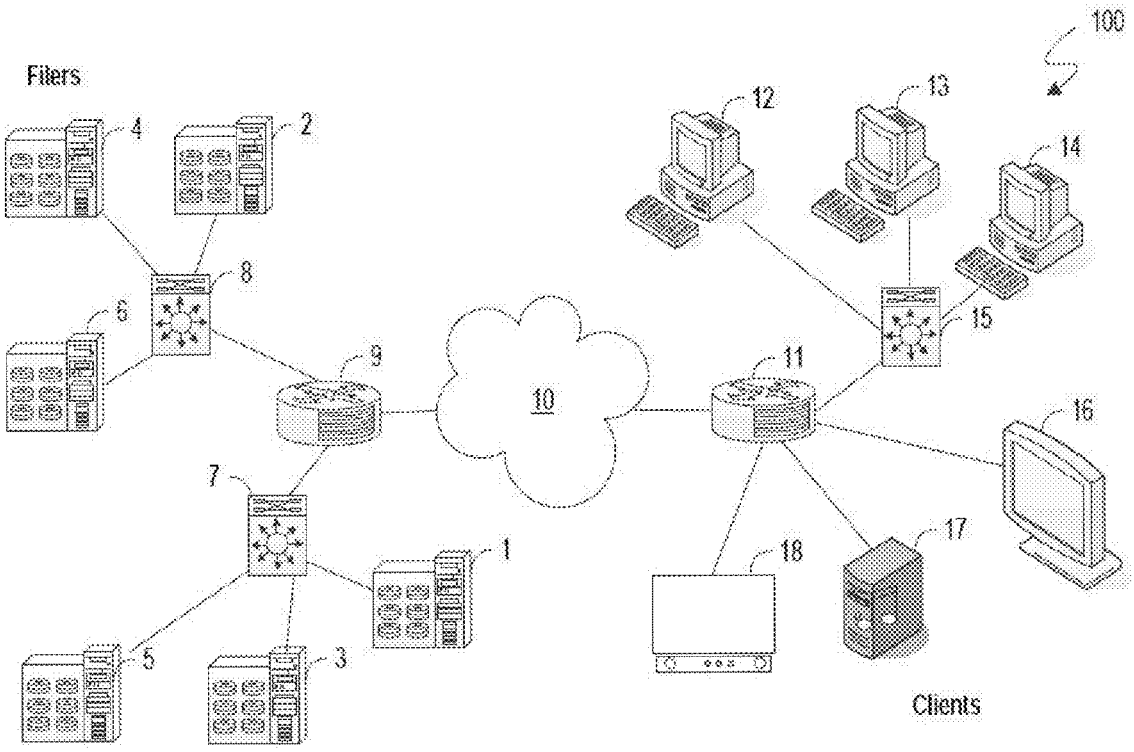
Publication Classification

(51) **Int. Cl.**
G06F 12/08 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 12/0891*** (2013.01); ***G06F 12/0815*** (2013.01)
USPC **711/135**; 711/141

(57) **ABSTRACT**

A network attached storage (NAS) caching appliance, system, and associated method to detect out-of-band accesses to a networked file system.



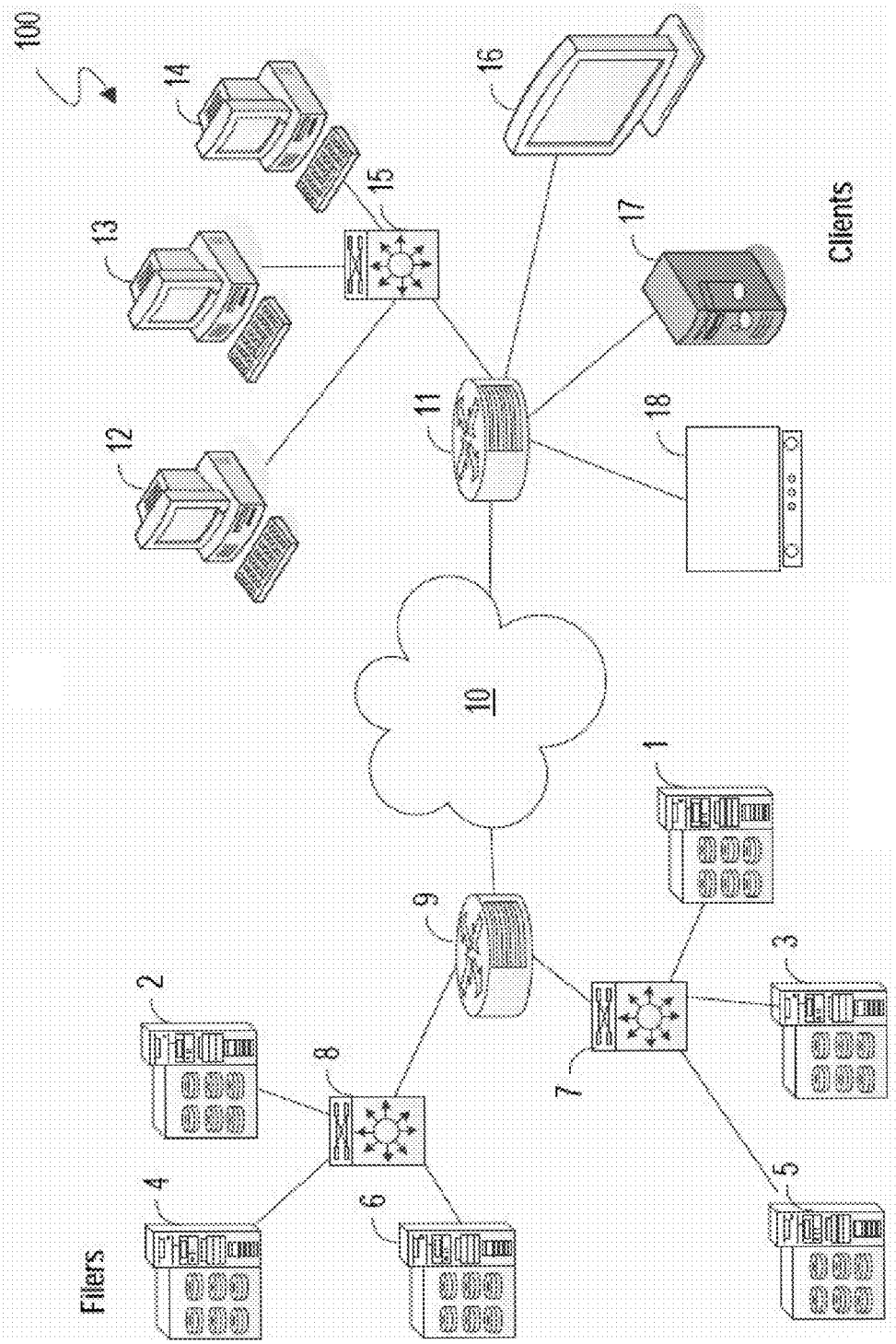


FIG. 1

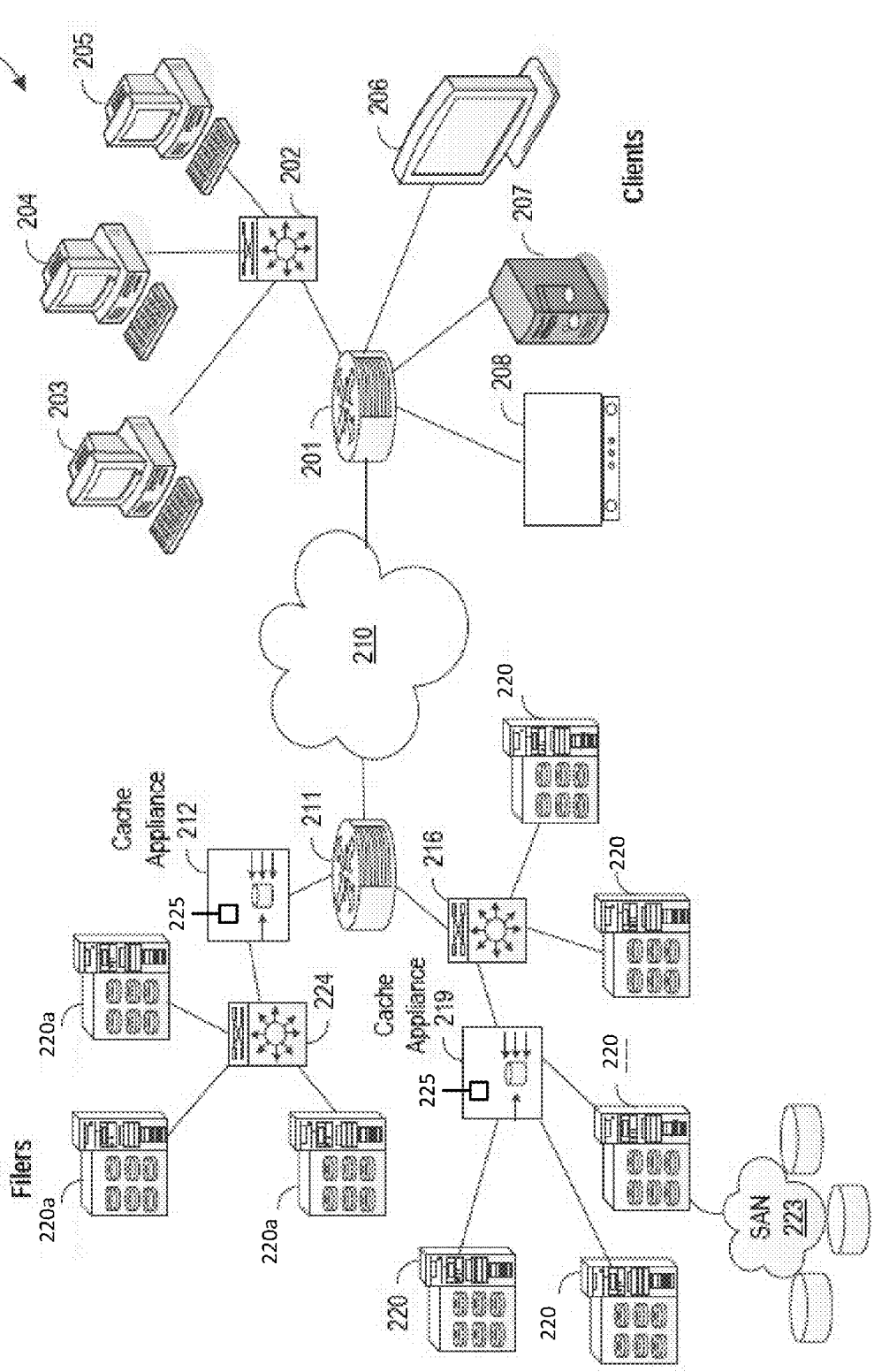

200
205
204
203
202
206
207
208
Clients
201
210
Cache Appliance
212
211
225
216
220
224
220a
Filers
Cache Appliance
219
SAN
223

FIG. 2

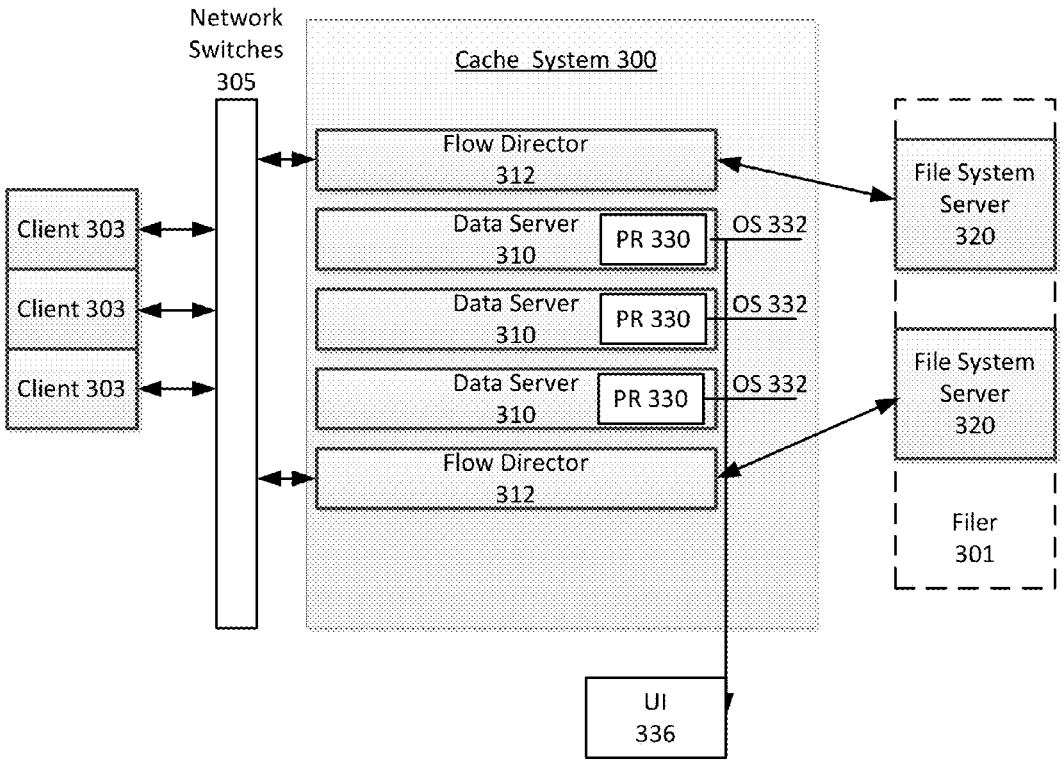

Network
Switches
305
Cache System 300
Flow Director
312
Data Server
310
PR 330
OS 332
Data Server
310
PR 330
OS 332
Data Server
310
PR 330
OS 332
Flow Director
312
Client 303
Client 303
Client 303
File System
Server
320
File System
Server
320
Filer
301
UI
336

FIG. 3

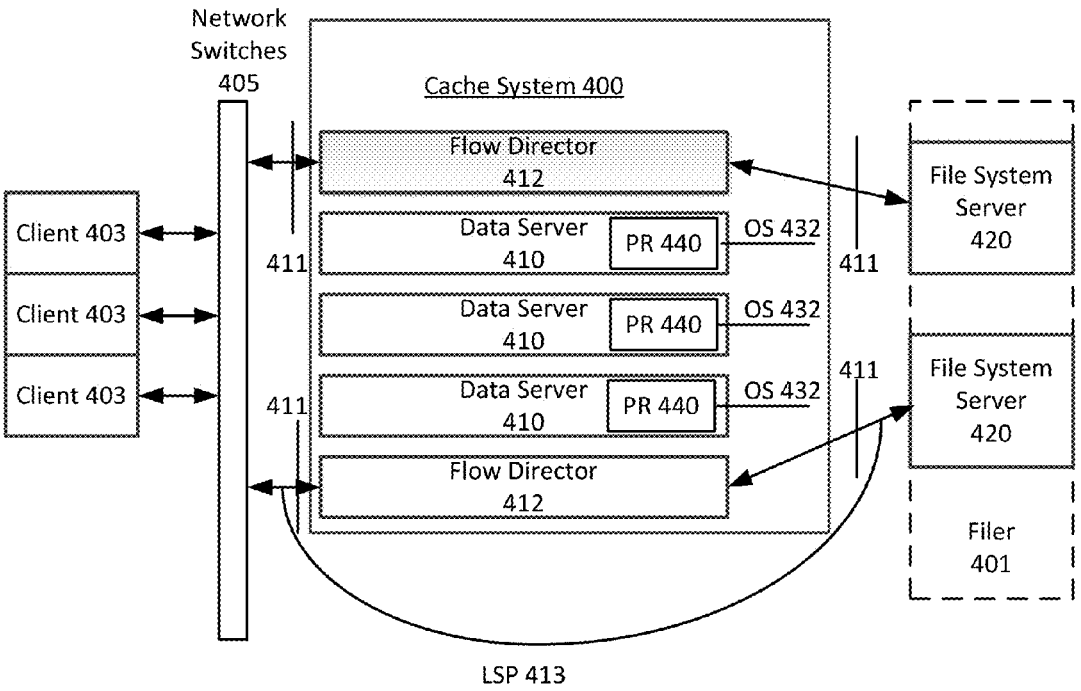

Network
Switches
405
Cache System 400
Flow Director
412
Data Server
410
PR 440
OS 432
Data Server
410
PR 440
OS 432
Data Server
410
PR 440
OS 432
Flow Director
412
Client 403
Client 403
Client 403
411
411
411
411
File System
Server
420
File System
Server
420
Filer
401
LSP 413

FIG. 4

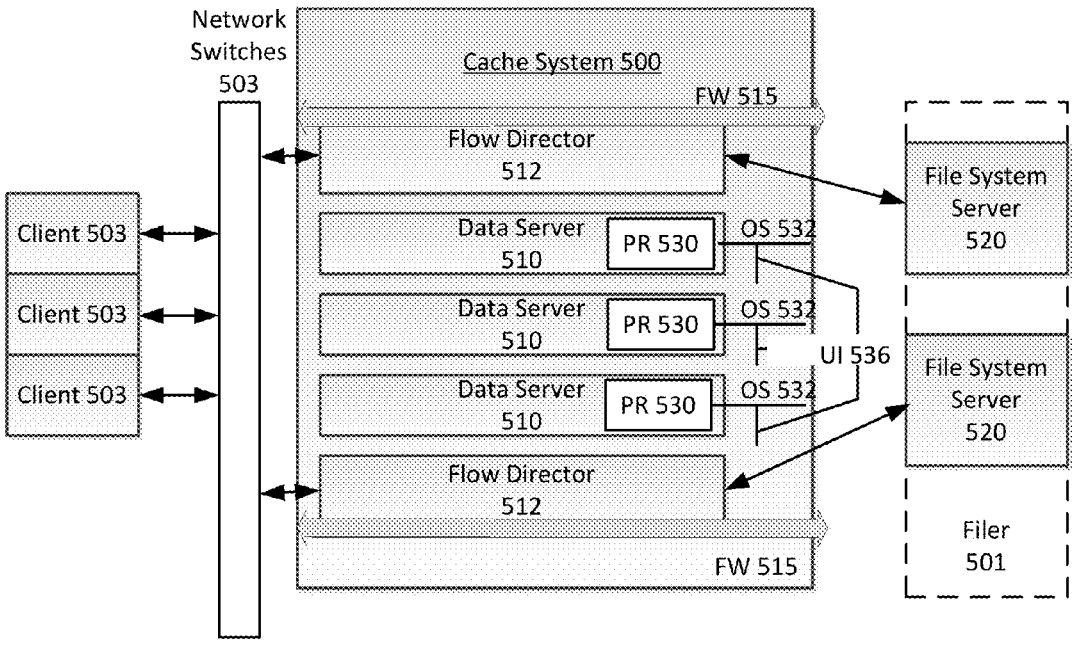

Network
Switches
503
Cache System 500
FW 515
Flow Director
512
Data Server
510
PR 530
OS 532
Data Server
510
PR 530
OS 532
Data Server
510
PR 530
OS 532
UI 536
Flow Director
512
FW 515
Client 503
Client 503
Client 503
File System
Server
520
File System
Server
520
Filer
501

FIG. 5

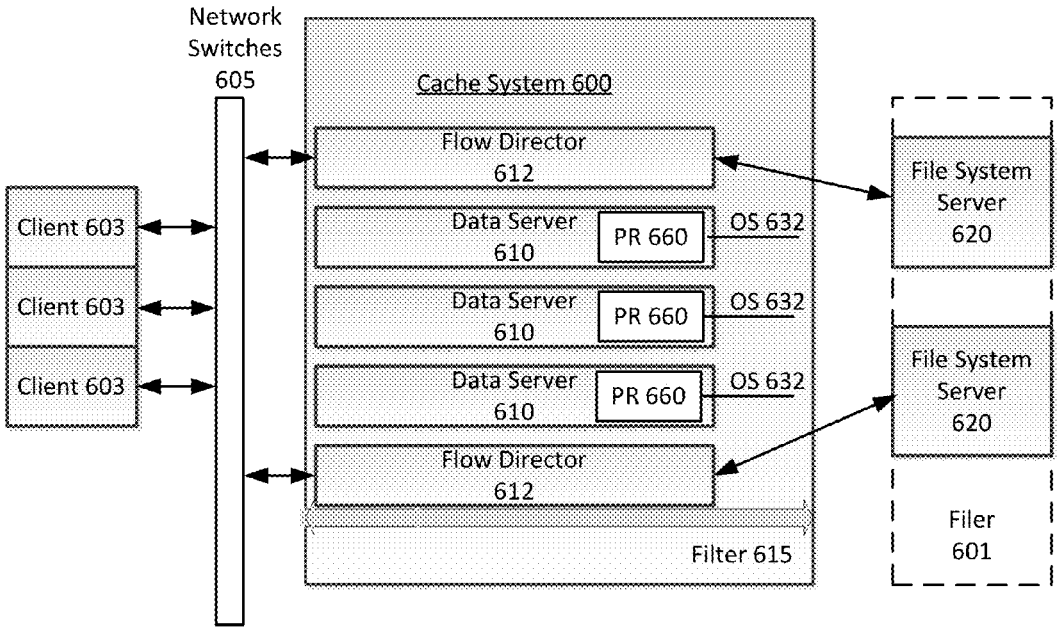

Network
Switches
605
Cache System 600
Flow Director
612
Data Server
610
PR 660
OS 632
Data Server
610
PR 660
OS 632
Data Server
610
PR 660
OS 632
Flow Director
612
Filter 615
Client 603
Client 603
Client 603
File System
Server
620
File System
Server
620
Filer
601

FIG. 6

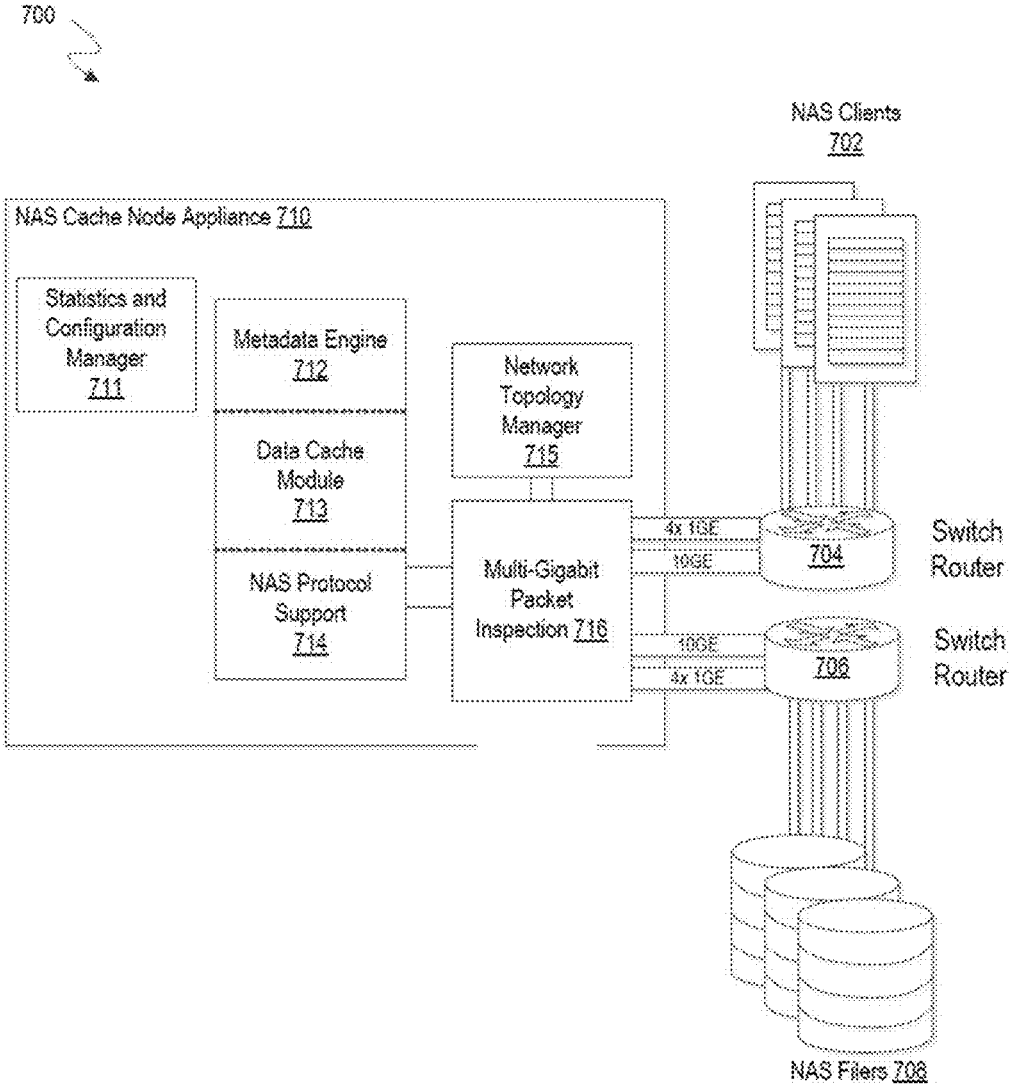

700
NAS Clients
702
NAS Cache Node Appliance 710
Statistics and Configuration Manager 711
Metadata Engine 712
Data Cache Module 713
NAS Protocol Support 714
Network Topology Manager 715
Multi-Gigabit Packet Inspection 716
4x 1GE
10GE
704
Switch Router
10GE
4x 1GE
706
Switch Router
NAS Filers 708

FIG. 7

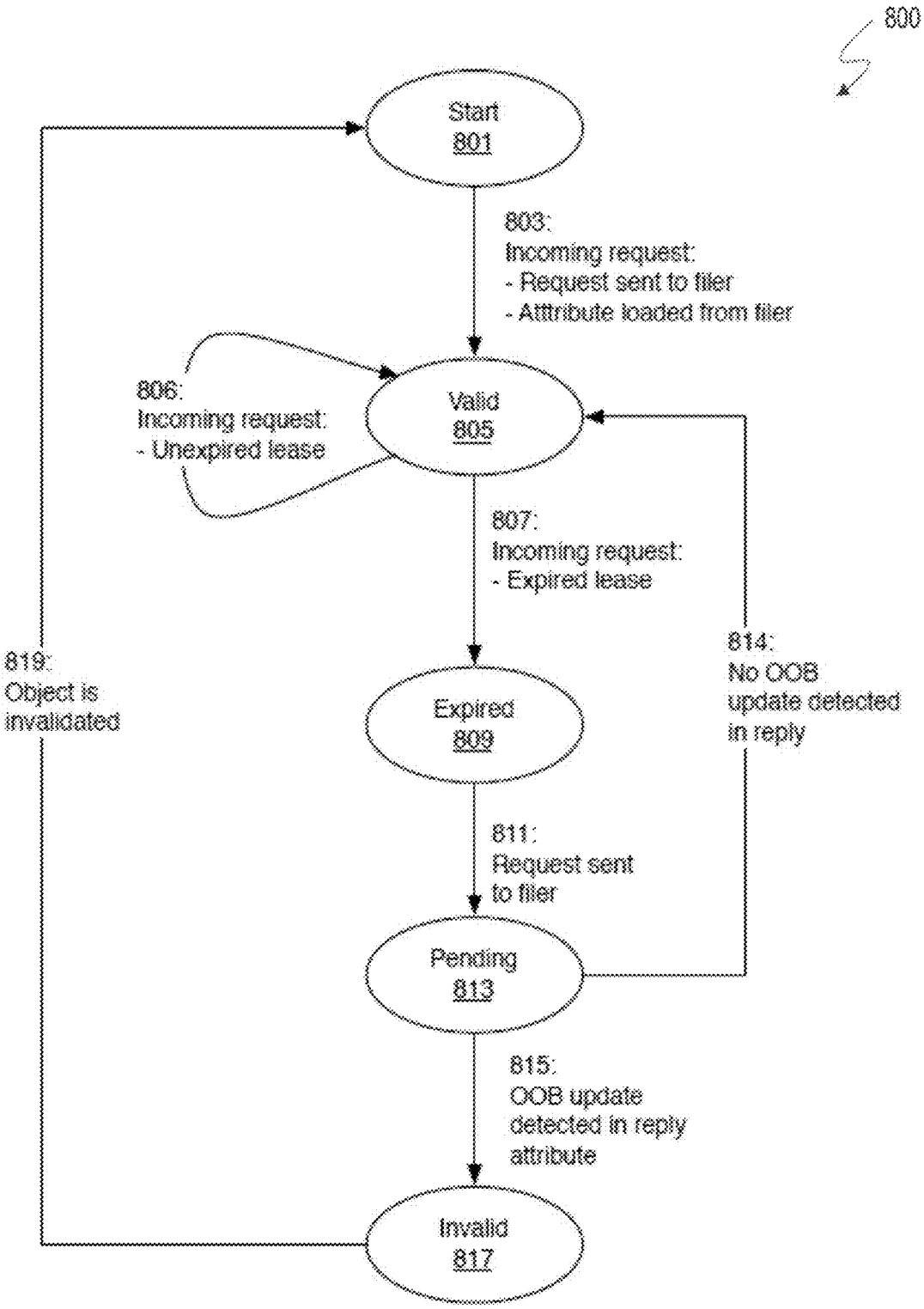

800
Start
801
803:
Incoming request:
- Request sent to filer
- Atttribute loaded from filer
806:
Incoming request:
- Unexpired lease
Valid
805
807:
Incoming request:
- Expired lease
819:
Object is
invalidated
Expired
809
814:
No OOB
update detected
in reply
811:
Request sent
to filer
Pending
813
815:
OOB update
detected in reply
attribute
Invalid
817

FIG. 8

DETECTION OF OUT-OF-BAND ACCESS TO A CACHED FILE SYSTEM

RELATED APPLICATIONS

[0001] This patent application claims benefit of priority to Provisional U.S. Patent Application No. 61/702,692; the aforementioned priority application being hereby incorporated by reference in its entirety for all purposes.

TECHNICAL FIELD

[0002] Examples described herein relate to detection of out-of-band access to a cached file system.

BACKGROUND

[0003] Data storage technology over the years has evolved from a direct attached storage model (DAS) to using remote computer storage models, such as Network Attached Storage (NAS) and a Storage Area Network (SAN). With the direct storage model, the storage is directly attached to the workstations and application servers, but this creates numerous difficulties with the administration, backup, compliance and maintenance of the directly stored data. These difficulties are alleviated at least in part by separating the application server/workstations from the storage medium. For example, FIG. **1** depicts a typical NAS system **100** in which a number of PCs, workstations and application servers (clients) use a network **10** to access storage resources on a number of remote network attached storage and file servers (or filers). In the depicted system **100**, each of the networked PC or workstation devices **12-14** and application servers **16-18** may act as a storage client that is connected to the network **10** by the appropriate routers **11** and switches **15** to remotely store and retrieve data with one or more NAS filers **1-6**, which in turn are connected to the network **10** by the appropriate routers **9** and switches **7-8**. Typically, the storage clients (e.g., **14**) use an IP-based network protocol, such as CIFS and NFS, to communicate store, retrieve and modify files on an NAS filer (e.g., **5**).

[0004] Conventional NAS devices are designed with data storage hardware components (including a plurality of hard disk drives, one or more processors for controlling access to the disk drives, I/O controller and high speed cache memory) and operating system and other software that provides data storage and access functions. Even with a high speed internal cache memory, the access response time for NAS devices continues to be outpaced by the faster processor speeds in the client devices **12-14**, **16-18**, especially where anyone NAS device may be connected to a plurality of clients. In part, this performance problem is caused by the lower cache hit rates that result from a combination of larger and constantly changing active data sets and large number of clients mounting the NAS storage device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIG. **1** depicts a prior art NAS system.

[0006] FIG. **2** illustrates an example of a networked system that utilizes intelligent, cache appliances, including topology detection logic, according to an embodiment.

[0007] FIG. **3** illustrates an example of a cache system for use with a system such as described with FIG. **2**.

[0008] FIG. **4** illustrates another example of a cache system for use with a system such as described with FIG. **2**.

[0009] FIG. **5** illustrates another example of a cache cluster for use with a system such as described with FIG. **2**.

[0010] FIG. **6** illustrates another example of a cache system for use with a system such as described with FIG. **2**.

[0011] FIG. **7** illustrates an architecture network, according to one or more embodiments.

[0012] FIG. **8** illustrates a request state diagram for handling out-of-band update requests at a cache node appliance, according to one or more embodiments.

DETAILED DESCRIPTION

[0013] Some embodiments described herein include a network attached storage (NAS) caching appliance, system, and associated method to detect out-of-band accesses to a networked file system.

[0014] According to some embodiments, a network attached storage (NAS) caching system is provided that delivers enhanced performance to I/O intensive applications while relieving overburdened storage subsystems. In some embodiments, a caching solution identifies active data sets in a networked file system, and uses predetermined policies to control what data gets cached using a combination of memory resources (e.g., DRAM and SSDs). Among other benefits, some examples provided herein improve performance by guaranteeing the best performance for the most important applications. When positioned between the storage clients and the networked file system, a caching system can intercept requests between the clients and filers and provides read and write cache acceleration by storing and recalling frequently used information.

[0015] In addition, embodiments described herein include a caching system that detects out-of-band operations that affect a networked file system. In some embodiments, the cache system detects out-of-band changes to the networked file system by comparing locally cached rnetadata with corresponding rnetadata from a NAS data storage device.

[0016] In some embodiments, a NAS cache appliance includes a multi-path detection functionality which compares cached rnetadata with corresponding rnetadata from the filer using a predetermined comparison triggering mechanism (e.g., defined lease times, on-demand probes, etc.) to ensure that NAS requests are serviced with correct content. In addition, a computer program product may be implemented that includes a non-transitory computer readable storage medium having computer readable program code embodied therein with instructions which are adapted to be executed to implement a method for operating a NAS caching appliance, substantially as described hereinabove. In selected embodiments, the operations described herein may be implemented using, among other components, one or more processors that run one or more software programs or modules embodied in circuitry and/or non-transitory storage media device(s) (e.g., RAM, ROM, flash memory, etc.) to communicate to receive and/or send data and messages. Thus, it will be appreciated by one skilled in the art that the present invention may be embodied in whole or in part as a method, system, or computer program product. For example, a computer-usable medium embodying computer program code may be used, where the computer program code comprises computer executable instructions configured to compare locally cached metadata/attributes with metadata/attributes received from the filer to detect out-of-band operations. Accordingly, the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally

be referred to herein as a "circuit," "module" or "system." Furthermore, the present invention may take the form of a computer program product on a computer-usable storage medium having computer-usable program code embodied in the medium.

[0017] Examples described herein provide for a high-performance network attached storage (NAS) caching appliance and system. In an embodiment, a NAS cache appliances manages the interconnect busses connecting one or more flow directors and cache node appliances, in order to monitor and respond to system health events/changes. In some embodiments, each of the NAS cache appliances includes an interconnect bus manager that provides address configuration and monitoring functions for each NAS cache appliance. In addition, a computer program product is disclosed that includes a non-transitory computer-readable storage medium having computer-readable program code embodied therein with instructions which are adapted to be executed to implement a method for operating a NAS caching appliance, substantially as described hereinabove. In selected embodiments, the operations described herein may be implemented using, among other components, one or more processors that run one or more software programs or modules embodied in circuitry and/or non-transitory storage media device(s) (e.g., RAM, ROM, flash memory, etc.) to communicate to receive and/or send data and messages. Thus, it will be appreciated by one skilled in the art that the present invention may be embodied in whole or in part as a method, system, or computer program product. For example, a computer-usable medium embodying computer program code may be used, where the computer program code comprises computer executable instructions configured to use the interconnect bus to monitor appliance failures using gratuitous ARP or heartbeat messages and respond to any failures at the interconnect bus or other system appliance. Accordingly, the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "circuit," "module" or "system." Furthermore, the present invention may take the form of a computer program product on a computer-usable storage medium having computer-usable program code embodied in the medium.

[0018] Among other benefits, a high-performance network attached storage (NAS) caching appliance can be provided for a networked file system to deliver enhanced performance to I/O intensive applications, while relieving overburdened storage subsystems. The examples described herein identify the active data sets of the networked system and use predetermined policies to control what data gets cached using a combination of DRAM and SSDs to improve performance, including guaranteeing the best performance for the most important applications. Examples described herein can further be positioned between the storage clients and the NAS filers, to intercept requests between the clients and filers, and to provide read and write cache acceleration by storing and recalling frequently used information. In some embodiments, a cache system that includes NAS caching appliance manages the network topology in which it is connected by dynamically probing the network to build a topology map of all accessible network devices. Using the topology map, the NAS cache appliances respond only when it is correct to do so, thus protecting against frame flooding while enabling minimal customer configuration.

[0019] In selected embodiments, the operations described herein may be implemented using, among other components, one or more processors that run one or more software programs or modules embodied in circuitry and/or non-transitory storage media device(s) (e.g., RAM, ROM, flash memory, etc.) to communicate to receive and/or send data and messages. Thus, it will be appreciated by one skilled in the art that the present invention may be embodied in whole or in part as a method, system, or computer program product. For example, a computer-usable medium embodying computer program code may be used, where the computer program code comprises computer executable instructions configured to provide dynamically detect and select file servers associated with a requested caching operation. Accordingly, the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "circuit," "module" or "system." Furthermore, the present invention may take the form of a computer program product on a computer-usable storage medium having computer-usable program code embodied in the medium.

[0020] It should be understood that as used herein, terms such as coupled, connected, electrically connected, in signal communication, and the like may include direct connections between components, indirect connections between components, or both, as would be apparent in the overall context of a particular embodiment. The term coupled is intended to include, but not be limited to, a direct electrical connection.

[0021] FIG. **2** illustrates an example of a networked system that utilizes intelligent, cache appliances, including out-of-band detection logic, according to an embodiment. In an example of FIG. **2**, an enterprise network system **200** includes multiple file system servers **220** and file system server groups **220***a* that collectively operate as one or more NAS filers of the enterprise file system **200**. The system **200** includes one or more cache appliances **212**, **219** located in front of a file system server **220** and/or file system server groups **220***a*. One or more clients **203**-**205** or **206**-**208** connect to and utilize the enterprise file system **200**. In the example provided, clients **203**-**205** correspond to, for example, mobile or desktop PCs or workstations, and clients **206**-**208** correspond to application servers (collectively termed "clients **203**-**208**"). Each of the clients **203**-**208** may run a separate application which requires access to remotely-stored application data. In operation, a requesting client sends a read or write request over the network **210** using the appropriate routers **201**, **211** and/or switches **202**, **216**, **224**. Such requests may be directed to the destination NAS filer using an appropriate IP-based network protocol, such as, for example, CIFS or NFS.

[0022] According to examples described herein, the cache appliances **212**, **219** are disposed logically and/or physically between at least some clients **203**-**208** and the file system server **220** and/or filer server groups **220***a* of the NAS filer. In more detail the cache appliances **212**, **219** include intelligent cache appliances which are installed in-line between individual clients **203**-**208** and the destination NAS filer. The individual clients **203**-**208** issue requests for a respective NAS filer provided with the system **200**. Such requests can include read or write requests in which file system objects of

the respective NAS filer is used. More specifically, examples described herein provide for the cache appliances **212**, **219** to (i) store a segment of the data of the NAS filer, and (ii) process requests from the clients **203-208** directed to the NAS filer. The cache appliances **212**, **219** can each include programmatic resources to optimize the handling of requests from the clients **203-208** in a manner that is transparent to the clients **203-208**. In particular, the cache appliances **212**, **219** can respond to individual client requests, including (i) returning up-to-date but cached application data from file system objects identified from the client requests, and/or (ii) queuing and then forwarding, onto the NAS filer, write, modify or create operations (which affect the NAS filer), and subsequently updating the contents of the respective cache appliances **212**, **219**. In general, the cache appliances **212**, **219** enable the individual client requests to be processed more quickly than would otherwise occur if the client requests were processed from the disk arrays or internal cache memory of the file system servers. More generally, the cache appliances **212**, **219** can be positioned in-line to cache the NAS filer without requiring the clients **203-208** to unmount from the NAS filer.

[0023] In an example of FIG. **2**, each cache appliance **212**, **219** can include one or more cache appliances that are connected together and working in tandem to form a single homogeneous caching device. Examples of cache appliances **212**, **219** are provided with embodiments described with FIG. **3** through FIG. **6**, as well as elsewhere in this application. Furthermore, in an example of FIG. **2**, each cache appliance **212**, **219** can include an appliance that is constructed as a high-speed packet processor with a substantial cache memory. For example, each cache appliance **212**, **219** can correspond to an appliance that includes a set of network processing resources (such as a network switch and network processor(s)), a dynamic cache memory, a non-volatile cache memory and/or cache controller(s). The processing resources of the individual cache appliances **212**, **219** can be configured to handle, for example, NFS type requests from the clients **203-208**.

[0024] As further shown by an example of FIG. **2**, individual cache appliances **212**, **219**, can be installed in multiple different locations of the system **200**. In this manner, the individual cache appliances **212**, **219** provide caching resources for one or more NAS filers, as shown by the placement of the cache appliance **219** in relation to file servers **220**, or alternatively, to a group of NAS filers as shown by the placement of the cache appliance **212** in relation to the NAS filers provided by the file servers **220** and file server groups **220***a*. However positioned, the cache appliances **212**, **219** each operate to intercept requests between the clients and the servers **220**. In this way, the cache appliances **212**, **219** are able to provide read and write cache acceleration by storing and recalling frequently used information. In some embodiments, the cache appliances **212**, **219** are positioned as part of a required path between a respective file server and some or all of the clients. In particular, the cache appliances **212**, **219** are positioned to intercept traffic directed from clients **203-208** to a particular file server **220** or set of file servers **220***a* in order to avoid cache coherency problems. In particular, cache coherency problems can arise when a piece of information stored with cache appliance **212**, **219** is modified through an alternate path.

[0025] As described with some examples, each cache appliance **212**, **219** can be provided with packet inspection functionality. In this way, each cache appliance **212**, **219** are able to inspect the information of each of the intercepted packets in each of the TCP/IP stack layers. Through packet inspection, cache appliances **212**, **219** can determine (i) the physical port information for the sender and receiver from the Layer 2 (data link layer), (ii) the logical port information for the sender and receiver from the Layer 3 (network layer), (iii) the TCP/UDP protocol connection information from the Layer 4 (transport layer), and (iv) the NSF/CIFS storage protocol information from the Layer 5 (session layer). Additionally, some embodiments provide that the cache appliances **212**, **219** can perform packet inspection to parse and extract the fields from the upper layers (e.g., Layer 5-Layer 7). Still further, some embodiments provide that the packet inspection capability enables each cache appliance **212**, **219** to be spliced seamlessly into the network so that it is transparent to the Layer 3 and Layer 4 layers.

[0026] According to embodiments, the cache appliances **212**, **219** can accelerate responses to storage requests made from the clients. In particular, the packet inspection capability enables each cache appliance **212**, **219** to be spliced seamlessly into the network so that it is transparent to the Layer 3 and Layer 4 layers and only impacts the storage requests by processing them for the purposes of accelerating them, i.e., as a bump-in-the-wire. Rather than splicing all of the connection parameters in the Layer 2, Layer 3 and Layer 4, some embodiments provide that each cache appliance **212**, **219** can splice only the connection state, source sequence number and destination sequence number in Layer 4. By leaving unchanged the source and destination MAC addresses in the Layer 2, the source and destination IP addresses in the Layer 3 and the source and destination port numbers in the Layer 4, the cache appliances **212**, **219** can generate a programmatic perception that a given client **203-208** is communicating with one of the NAS filers of the enterprise network system **200**. As such, there is no awareness at either the clients **203-208** or file servers **220**, **220***a* of any intervening cache appliance **212**, **219**. In this way, the cache appliances **212**, **219** can be inserted seamlessly into an existing connection with the clients **203**, **208** and the NAS filer(s) provided with the system **200**, without requiring the clients to be unmounted. Additionally, among other benefits, the use of spliced connections in connecting the cache appliances **212**, **219** to the file servers **220** and file server groups **220** enable much, if not all, of the data needs of the individual clients to be served from the cache, while providing periodic updates to meet the connection timeout protocol requirements of the file servers **220**.

[0027] In more detail, the cache appliance **212**, **219** can process a read or write request by making only Layer 1 and Layer 2 configuration changes during installation or deployment. As a result, no filer or client configuration changes are required in order to take advantage of the cache appliance. With this capability, an installed cache appliance **212**, **219** (e.g., appliance) provides a relatively fast and transparent storage caching solution which allows the same connections to be maintained between clients and filers. As described with some embodiments, if there is a failure at the cache appliance **212**, **219**, the cache appliance automatically becomes a wire (e.g., pass through) between the client and filer who are able to communication directly without any reconfiguration.

[0028] According to some embodiments, cache appliance **212**, **219** are implemented as a network attached storage (NAS) cache appliance, and connected as an in-line appliance or software that is positioned in the enterprise network system

200 to intercept requests to one or more of the file servers **220**, or server groups **220***a*. This configuration provides clients **203-208** expedited access to the data within the requested files, so as to accelerate NAS storage performance. As an appliance, cache appliances **212**, **219** can provide acceleration performance by storing the data of the NAS filers (provided from the file servers **220** and server groups **220***a*) in high-speed media. In some embodiments, cache appliances **212**, **219** are transparently installed appliances, deployed between the clients **203-208** and file system servers **220**, **220***a* without any network or reconfiguration of the endpoints. Without client or file server configuration changes, the cache appliances **212**, **219** can operate intelligently to find the active dataset (or a designated dataset) of the NAS filers, and further to copy the active data sets into DRAM and SSD memory. The use of DRAM and SSD memory provides improvement over conventional type memory used by the file servers. For example, in contrast to conventional approaches, embodiments described herein enable cache appliances **212**, **219** to (i) operate independently, (ii) operate in a manner that is self-contained, (iii) install in-line in the network path between the clients and file servers. Knowing the contents of each packet allows data exchanged with the file servers **220**, **220***a* (e.g., NFS/CIFS data) to be prioritized optimally the first time the data is encountered by the cache appliances, rather than being moved after-the-fact.

[0029] As described with an example of FIG. **7** and FIG. **8**, each of cache appliance **212**, **219** includes out-of-band detection logic **225**. The out-of-band detection logic **225** can perform operations to detect out-of-band changes to the system **200**. By detecting the out-of-band changes to the system **200**, the cache appliances **212**, **219** can, for example, ensure coherency is maintained between the networked file system and the cache resources.

[0030] FIG. **3** illustrates an example of a cache system for use with a system such as described with FIG. **2**. In particular, FIG. **3** illustrates a cache system **300** that includes multiple data servers **310** and flow directors **312**. In this way, the cache system **300** can include multiple appliances, including NAS cache appliances. The cache system **300** utilizes network switches **305** to connect to clients **303** across one or more networks. In implementation, the components of the cache system **300** (e.g., data servers **310**, flow directors **312**) can be positioned in-line with respect to clients **303** and file system servers **320** of a networked system **301**. Accordingly, connectivity between the clients **303** and the cache system **300**, as well as between the cache system **300** and the file system servers **320** of the networked system **301**, can be across one or more networks. The networked system **301** can correspond to, for example, a combination of file system servers of the networked system, as described with an example of FIG. **2** (e.g., see network system **200** of FIG. **2**).

[0031] According to one aspect, the cache system **300** includes one or more data servers **310**, one or more flow directors **312**, and processing resources **330**. In some implementations, the processing resources **330** that coincide with resources of the data servers **310** implement a cache operating system **332**. Additionally, the processing resources **330** can perform various analytic operations, including recording and/or calculating metrics pertinent to traffic flow and analysis.

[0032] In some embodiments, the data server **310** implements operations for packet-inspection, as well as NFS/CIFS caching. Multiple data servers **310** can exist as part of the cache system **300**, and connect to the file servers **320** of the networked system **301** through the flow director(s) **312**. The flow director(s) **312** can be included as active and/or redundant devices to interconnect the cache system **300**, so as to provide client and file server network connectivity for filer **301**.

[0033] The cache operating system **332** can synchronize the operation of the data servers **310** and flow directors **312**. In some embodiments, the cache operating system **332** uses active heartbeats to detect node failure (e.g., failure of one of the data servers **310**). If a node failure is detected, the cache operating system **332** removes the node from the cache system **300**, then instructs remaining nodes to rebalance and redistribute file responsibilities. If a failure is detected from one of the flow directors **312**, then another redundant flow director **312** is identified and used for redirected traffic.

[0034] In one implementation, a user interface **336** can be implemented through the processing resources **330**. The user interface **336** can be implemented as, for example, a web-interface. The processing resources **330** can be used to gather and view statistics, particularly as part of the operations of the data server **310** and the flow director **312**. The user interface **336** can be used to display metrics and statistics for purpose of, for example, troubleshooting storage network issues, and configuring the NAS cache system **300**. For example, administrators can use the user interface **336** to view real-time information on cache performance, policy effectiveness, and application, client, and file server performance.

[0035] According to some embodiments, the data servers **310** include packet inspection and NFS/CIFS caching infrastructure for the cache system **300**. In one implementation, the data servers **310** utilize multiple cache media to provide different performance levels. For example, in some embodiments, each data server **310** supports DDR3 DRAM and high performance SSD storage for caching. In operation, data servers **310** communicate with both clients **303** and file system servers **320**, by, for example, inspecting every message and providing the information necessary to intelligently cache application data.

[0036] In some embodiments, the data servers **310** can be implemented in a manner that is extensible, so as to enable expansion and replacement of data servers **310** from the cache system **300**. For example, each data server **310** can employ hot swappable power supplies, redundant fans, ECC memory and enterprise-level Solid State Disks (SSD).

[0037] Further, in some embodiments, the flow directors **312** operate as an enterprise-level Ethernet switch (e.g., 10 GB Ethernet switch). The flow directors **312** can further be implemented with software so as to sit invisibly between clients **303** and file system servers **320**. In the cache system **300**, the flow director **312** load balances the data severs **310**. The individual flow directors **312** can also provide the ingress and egress point to the network. Additionally, the flow directors **312** can also filter traffic that passes through non-accelerated protocols. In some implementations, flow directors **312** work in concert with the operating system **332** to provide failover functionality that ensures access to the cached data is not interrupted.

[0038] In some embodiments, the flow directors **312** can also operate so that they do not participate in switching protocols between client and file server reciprocal ports. This allows protocols like Spanning Tree (STP) or VLAN Trunking Protocol (VTP) to pass through without interference. Each flow director **312** can work with the data servers **310** in order to support, for example, the use of one or more of Link

Aggregation (LAG) protocols, 802.1Q VLAN tagging, and jumbo frames. Among other facets, the flow directors **312** can be equipped with hot swappable power supplies and redundant fans. Each flow director **312** can also be configured to provide active heartbeats to the data servers **310**. In the event that one of the flow directors **312** becomes unresponsive, an internal hardware watchdog component can disable client/file server ports in order to facilitate failover on connected devices. The downed flow director **312** can then be directed to reload and can rejoin the cache system **300** if once again healthy.

[0039] FIG. **4** illustrates another example of a cache system for use with a system such as described with FIG. **2**. In particular, FIG. **4** illustrates a cache system **400** that includes multiple data servers **410**, flow directors **412** and processing resources **430** on which an operating system **432** can be implemented. In this way, the cache system **300** can include multiple appliances, including NAS cache appliances. The cache system **400** utilizes network switches **405** to connect to clients **403** across one or more networks. In implementation, the cache system **400** can be positioned in-line with respect to clients **403** and file system servers **420** of a networked system **401**. Accordingly, connectivity between the clients **403** and the cache system **400**, as well as between the cache system **400** and the file system servers **420** of the networked system **401**, can be across one or more networks. As with an example of FIG. **3**, the networked system or filer **401** can correspond to, for example, a combination of file system servers **420** that provide one or more NAS filers, as described with an example of FIG. **2** (e.g., see system **200** of FIG. **2**).

[0040] In an example of FIG. **4**, the flow directors **412** and data server **410** support **802**.1Q VLAN tagging connections **411** to the client-side switch and the file servers. The data servers **410** operate to maintain the connection state between the clients **403** and file servers **420** of the filer, so that network traffic can flow indiscriminately through either of the flow directors **412**. In this way, the flow directors **412** are essentially equal bidirectional pathways to the same destination. As a result, any link failover is negotiated between the client switch and individual file servers, with the operating system **432** facilitating failover with Link State Propagation (LSP) communications **413** and link aggregation protocols. In this arrangement, the flow director(s) **412** provide an LSP feature for the in-line cache system **400** to maintain end-to-end link state between the client switch and file server. Since, in the example provided with FIG. **4**, the flow director(s) **412** are physically located between these devices, these flow directors actively monitor reciprocal connections so both client-side and file server-side connections are in sync. This allows implementation of the LAG protocol (if employed) to dynamically adjust in case of link failure.

[0041] FIG. **5** illustrates another example of a cache cluster for use with a system such as described with FIG. **2**. In an example of FIG. **5**, an in-line NAS cache system **500** includes two (or more) flow directors **512**, a supporting data server **510**, and processing resources **530** on which an operating system **532** can be implemented. In this way, the cache system **500** can include multiple appliances, including NAS cache appliances. The cache system **500** utilizes network switches **505** to connect to clients **503** across one or more networks. In implementation, the cache system **500** can be positioned in-line with respect to clients **503** and file system servers **520** of a networked system **501**. Accordingly, connectivity between the clients **503** and the cache system **500**, as well as between the cache system **500** and the file system servers **520** of the networked system **501**, can be across one or more networks. As with an example of FIG. **3**, the networked system or filer **501** can correspond to, for example, a combination of file system servers **520** that provide one or more NAS filers, as described with an example of FIG. **2** (e.g., see system **200** of FIG. **2**).

[0042] The data servers **510** can be connected between individual file system servers **520** and a client-side switch for some of the clients **503**. As depicted, the flow directors **512** and data server **510** provide a fail-to-wire pass through connection **515**. The connection **515** provides a protection feature for the in-line cache system **500** in the event that the data servers **510** fail to maintain heartbeat communications. With this feature, the flow director(s) **512** are configured to automatically bypass the data server(s) **510** of the cache system in case of system failure. When bypassing, the flow directors **512** send traffic directly to the file system servers **520**. Using active heartbeats, the flow directors **512** can operate to be aware of node availability and redirect client requests to the file system server **520** when trouble is detected at the cache system.

[0043] A bypass mode can also be activated manually through, for example, a web-based user interface **536**, which can be implemented by the processing resources **530** of the cache system **500**. The active triggering of the bypass mode can be used to perform maintenance on data server nodes **510** without downtime. When the administrator is ready to reactivate the cache system **500**, cached data is revalidated or flushed to start with a "clear cache" instruction.

[0044] FIG. **6** illustrates another example of a cache system for use with a system such as described with FIG. **2**. In an example of FIG. **6**, an in-line cache system **600** includes two (or more) flow directors **612** and one or more supporting data servers **610**. In this way, the cache system **600** can include multiple appliances, including NAS cache appliances. The cache system **600** utilizes network switches **605** to connect to clients **603** across one or more networks. The data server **610** can be connected between one of the file system servers **620** of the NAS filer **601** and clients **603** (including iSCSI clients). In implementation, the cache system **600** can be positioned in-line with respect to clients **603** and file system servers **620** of a networked system **601**. Accordingly, connectivity between the clients **603** and the cache system **600**, as well as between the cache system **600** and the file system servers **620** of the networked system **601**, can be across one or more networks. As with an example of FIG. **3**, the networked system or filer **601** can correspond to, for example, a combination of file system servers **620** that provide one or more NAS filers, as described with an example of FIG. **2** (e.g., see system **200** of FIG. **2**).

[0045] As depicted, the flow directors **612** and data server **610** of the cache system **600** provide a low latency, wire-speed filtering feature **615** for the in-line cache system **600**. With filtering feature **615**, the flow director(s) **612** provide advanced, low-latency, wire-speed filtering such that the flow director filters only supported-protocol traffic to the system. Substantially all (e.g., 99%) other traffic is passed straight to the file system servers **620** of the NAS filer **601**, thereby ensuring that the data servers **610** focus only on traffic that can be cached and accelerated.

[0046] In support of the various features and functions described herein, each cache system **600** implements operating system **632** (IQ OS) (e.g., FreeBSD) to be customized

with a purpose built caching kernel. Operating across all data servers and interacting with flow directors in the cache system, the OS **632** serves basic functions that include network proxy, file object server, and generic storage access. As a network proxy between clients and file servers, the OS **632** performs Layer 2 topology discovery to establish what is physically connected. Once the topology is determined, it maintains the network state of all connections. As requests are intercepted, the requests are converted to NAS-vendor independent file operations, streamlining the process while allowing the cache system **600** to incorporate other network protocols in the future.

[0047] Once requests are converted, the cache appliance system handles generic metadata operations, and data operations are mapped to virtual devices. Virtual devices can be implemented with DRAM, flash memory, and/or other media, and are categorized according to their performance metrics, including latency and bandwidth. Virtualization of devices allows the OS **632** to easily incorporate faster media to further improve performance or denser media to add cache capacity. Once the media hierarchy or tier is established within the cache resources of the system **600**, blocks are promoted and demoted based on frequency of use, unless "pinned" to a specific tier by the administrator. Additionally, in some implementations, the data servers **610** can operate a policy engine, which can implement user-defined polices, and proactively monitor the tiers of cache and prioritize the eviction of data blocks.

[0048] In one implementation, the cache system **600** may include a DRAM virtual tier where metadata is stored for the fastest random I/O access. In the DRAM virtual tier, user-defined profiles can be "pinned" for guaranteed, consistent access to critical data. SWAP files, database files, and I/O intensive virtual machine files (VMDKs) are a few examples of when pinning data in DRAM can provide superior performance.

[0049] In addition or in the alternative, some implementations provide that each cache system **600** may include a virtual tier for Solid State Disks (SSD) which can be added at any time to expand cache capacity. To maximize performance and capacity, individual SSDs are treated as an independent virtual tier, without RAID employment. In the event of a failed SSD, the overall cache size will shrink only by the missing SSD. The previously cached data will be retrieved from the file server (as requested) and stored on available media per policy.

[0050] Using packet inspection functionality of the data server **610**, the OS **632** at the cache system **600** learns the content of data streams, and at wire-speed, makes in-flight decisions based on default or user-defined policies to efficiently allocate high-performance resources where and when they are required most. Because data is initially stored to its assigned virtual tier, blocks are moved less frequently, which increases overall efficiency. However, as data demands change, the OS **632** also considers frequency of use to promote or demote blocks between tiers (or evict them completely out of cache).

[0051] In support of the caching operations, each cache system **600** can include one or more default built-in policies which assign all metadata to the highest tier (currently DRAM) and all other data to a secondary pool with equal weight. Frequency of use will dictate if data is to be migrated between tiers. And with no user-defined profiles enabled, the default policy controls caching operations. In addition, one or more file policies may be specified using filenames, file extensions, file size, file server, and file system ID (FSID) in any combination with optional exclusions. An example file policy would be to "cache all *.dbf files less that 2 GB from file server 192.168.2.88 and exclude file201.dbf." Client policies may also use IP addresses or DNS names with optional exclusions to specify cache operations. An example client policy would be to "cache all clients in IP range: 192.168.2.0/24 and exclude 192.168.2.31"

[0052] As will be appreciated, one or more cache policy modifiers may be specified, such as a "quota" modifier which imposes a limit on the amount of cache a policy consumes and can be specified by size or percent of overall cache. Quota modifiers can be particularly useful in multitenant storage environments to prevent one group from over-consuming resources. In addition, a "schedule" modifier may be used to define when a policy is to be activated or disabled based on a time schedule. An example, the cache system **600** can activate the "Nightly Software Build" profile at 9 pm and disable at 6 am. Another policy modifier referenced above is a user-created exception to "pin" data to a particular tier or the entire cache. A pinned policy means other data cannot evict the pinned data—regardless of frequency of use. Such a policy can be useful for data that may not be accessed often, but is mission-critical when needed. In busy environments that do not support pinning, important but seldom used data will never be read from cache because soon after it is cached, the data is evicted before it is needed again. Pinned policies can address this unwanted turnover. Yet another modifier is a "Don't Cache" modifier which designates by file name of client request selected data that is not to be cached. This option can be useful when dealing with data that is only read once, not critical, or which may change often. As another example, a "priority" modifier may be used to manually dictate the relative importance of policies to ensure data is evicted in the proper order. This allows user-defined priorities to assign quality of service based on business needs.

[0053] Using the cache policies and modifiers, the cache behavior of the cache system **600** can be controlled to specify data eviction, migration, and multi-path support operations. For example, the cache system **600** can make an eviction decision based on cache priority from lowest to highest (no cache, default, low, high, and pin), starting with the lowest and moving to higher priority data only when the tier is full. In one implementation, eviction from cache resources of the cache system **600** can be based on priority, and then usage. For example, the lowest priority with the least accessed blocks will be evicted from cache first, and the highest priority, most used blocks will be evicted last.

[0054] The cache system **600** can also control the migration of data within the cache based strictly by usage, so that the most active data, without regard to priority, will migrate to the fastest cache tier. Likewise, as other data becomes more active, stale data will be demoted. Data pinned to a specified tier is excluded from migration.

[0055] In some implementations, the cache system **600** can also include a Mufti-Path Support (MPS) mechanism for validating the data in the cache resources of the cache system **600**. With the MPS mechanism, the NAS cache checks backend file server attributes at a configurable, predefined interval (lease time). Data may change when snap-restoring, using multiprotocol volumes (i.e., CIFS, NFSv2/4), or if there are clients directly modifying data on the backend file server. When a client reads a file, MPS evaluates its cache lease time

to determine whether it needs to check file server attributes. If not expired, the read will be served immediately from cache. If expired, MPS checks the backend file server to confirm no changes have occurred. If changes are found, MPS will pull the data from the file server, send it to the client, reset its lease, and update the cache. With regular activity, file leases should rarely expire since they are updated on most NFS operations. Expiration only occurs on idle files. MPS timeout can be configured from, for example, a minimum (e.g., 3 seconds) to a maximum (e.g., 24 hours).

[0056] FIG. **7** illustrates a NAS architecture network **700** in which a plurality of filer systems **708** are connected across switch routers **706**, **704** to a plurality of remote clients **702** using an intermediate cache node appliance **710**. With such NAS architectures, the filer system **708** actively managing the data is considered its "owner" or "authoritative entity" because it is directly responsible for the state of the data set. In this role, the filer **708** provides services to remote clients **702** via NAS protocols, such as NFS and CIFS. These filer services include data storage and retrieval, naming, authorization (e.g., access control), transaction, and locking services. When a request to read or write application data is received from a storage client **702**, the NAS cache appliance/cluster **710** uses dedicated, high-speed packet inspection hardware **716** to inspect the packets of incoming requests to determine if they should be passed inward for further processing by the NAS cache appliance/cluster **710** or forwarded to another destination, such as a NAS filer **708**. For example, if the NAS client **702** requests application data that is stored on the NAS cache appliance/cluster **710**, the packet inspection hardware **716** may process the request based on I/O profiles to determine if the request is to be processed by the NAS cache appliance/cluster **710**. If so, the request is passed internally to the tiered memory cache system. For example, Tier 1 of NAS the cache appliance/cluster **710** may be reserved for the most critical data (including email, high transaction databases, business critical processes and line of business applications), while Tier 0 storage refers to an in-band, network-resident, policy-driven, high-performance, scalable tier of memory subsystems that is used for the storage of business critical data under control of a policy engine that is managed independently from the one or more NAS filers. Within the tiered memory, a volatile or dynamic random access memory virtual tier may be used to store metadata and/or application data for the fastest random I/O access, while a non-volatile random access memory (NVRAM) provides a space for caching pending write operations to NAS filers for the purpose of maintaining data coherency in a failure event. Such a failure event can correspond to incidents of network packets not arriving to their destination. If it is determined that the request cannot be serviced by the NAS cache appliance/cluster **710**, the client request is sent to the destination NAS **708**.

[0057] To support caching operations, the cache node appliance **710** can be provided as an external, active, NAS device that provides services similar to that of a filer **708** while deferring data set ownership to the filer, thereby acting as a proxy to the filer **708**. As shown in FIG. **7**, the cache node appliance **710** is positioned in the network **700** between the NAS clients **702** and filers **708** to service NAS requests on behalf of the filer **708**. Although it provides similar services to the filer **708**, it is not the authoritative entity—the filer remains the sole owner of the data set.

[0058] In addition to providing NAS protocol support **714** and data caching module **713**, the NAS cache appliance/cluster **710** includes a metadata engine (MDE) **712**. The MDE **712** uses metadata to detect out-of-band (OOB) operations relating to file system objects at the cached file system that are executed by the filer without knowledge of the NAS cache system. As will be appreciated, a file system object is a data object (e.g., file or directory) that resides on, and is managed by, a file system, while metadata is the meta information (e.g., file size, creation-time, and modification-time) that describes a file system object.

[0059] Among other functionality, the MDE **712** can provide for metadata storage and retrieval by caching file system object metadata. The MDE **712** can also provide metadata services by servicing and accelerating metadata requests, as well as naming services, such as providing lookup services for the clients. The MDE **712** can also provide authorization services to enforce access control, and transaction management services by coordinating concurrent requests. In addition, the MDE **712** may provide locking services on behalf of the filer.

[0060] The MDE **712** may also provide additional services when the cache node appliance **710** is deployed transparently. The MDE **712** can be inserted into an ongoing NAS conversation (i.e., a set of requests flowing between a NAS client and filer). Once inserted into the conversation, the MDE **712** proxies on behalf of the client **702** to the filer **708**, and on behalf of the filer **708** to the client **702**. In this capacity, the MDE **712** takes over the role of servicing, or forwarding, NAS requests as required. In servicing the requests, the MDE **712** maintains (data and metadata) consistency with the filer for the relevant file(s). And because the MDE **712** can be inserted into ongoing NAS traffic, the MDE **712** can maintain a sparse namespace so that it is not required to be inserted prior to the time the NAS client mounts the exported filer path.

[0061] By virtue of being located in the network **700** between a NAS client **702** and filer **708**, it is possible for out-of-band (OOB) requests to occur. In particular, OOB requests can occur when one or more NAS requests reach the filer **708** while bypassing the cache system's MDE **712**. As a result of an OOB request, a file can be modified at the filer **708** without the knowledge of the MDE **712**. Yet, the MDE, as a proxy to the filer **708**, must remain consistent with the authoritative entity or filer **708** in order to ensure that the NAS requests are serviced with the correct content, as if it were being serviced by the filer **708**. Consequently, embodiments implement the MDE **712** to detect out-of-band updates.

[0062] As disclosed herein, the MDE **712** may be configured to handle out-of-band updates using one or more detection schemes, depending on the applicable protocol. Under a first detection technique, the MDE **712** grants each file system object a shelf-life, or lease time, having a predefined duration such that after its expiration, the MDE probes the filer **708** to determine if out-of-band changes have occurred. If an OOB change has occurred, the MDE **712** invalidates its own copy of the specific object so that it can obtain the filer's version of the object.

[0063] According to another detection technique, the MDE **712** performs probing operations on demand to detect OOB changes. In other words, MDE **712** implements operations to remain consistent with the filer for those objects that are being accessed.

[0064] Under yet another detection technique, the MDE **712** is selective about what it caches and which NAS requests it services. For instance, MDE **712** can selectively ignore NAS requests for files that were opened before the MDE was inserted into the NAS stream.

[0065] The MDE **712** may also use an OOB request detection technique which probes the filer using existing, standard, NAS requests. In other words, it requires no proprietary "hooks" at the filer. Alternatively, the MDE **712** may leverage NAS protocol-specific information to implement its out-of-band detection. In other words, OOB detection can be implemented to be protocol specific. To provide an example implementation for a NAS architecture providing file access using the NFS Version 3 network file sharing protocol specification (NFSv3, RFC 1813), the MDE **712** may be configured to detect out-of-band updates by maintaining the metadata of the file in its cache, where the metadata represents the latest state of the file from the MDE's perspective. In this context, the metadata for NFS version 3 is referred to as attributes, which include the file size, the creation-time, and the modification-time, etc. Any unexpected divergence between the metadata on the filer **708** and the cached version at the cache node appliance **710** indicate an out-of-band update, which results in the invalidation of the cached content. The MDE **712** also applies a lease time on the cached metadata so that, whenever a NAS request is received, the MDE **712** checks the lease time of cached file metadata. If the lease time has expired, the MDE **712** forwards the request to the filer, atomically, and waits for a reply.

[0066] Embodiments recognize that with the NFS Version 3 protocol specification, replies from the filer **708** include the "post-op" attributes which describe the state of the file after a particular request has been processed or executed. In addition, NFSv3 update operations typically include metadata which are called the "pre-op" attributes. The pre-op attribute is a subset of the file's metadata, but includes all the elements necessary to describe the state of the file just before the request was executed. For query operations, such as LOOKUP, which do not modify the file system object and do not return the pre-op attributes, the MDE **712** compares the returned post-op attributes with what is cached. Any divergence indicates that an OOB update has occurred, and the object is consequently invalidated. For update operations, such as MKDIR, both the pre-op and post-op attributes are returned. In this case, the MDE compares the returned pre-op attributes with the cached version. If a difference is detected, then the object is invalidated.

[0067] In the NFS Version 3 protocol specification, the MDE **712** may also leverage the out-of-band detection mechanism to enable concurrent, non-overlapping, file write operations. For example, in the situation where there are two NFS_WRITE operations, the first one writes 1024 bytes at offset 0 of the file, and the second one writes 1024 at offset 2048 of the same file. Although both operations update different regions of the file, both operations also update a shared element of the file, namely, the attribute. This behavior introduces problems if the response is processed out of order since an out-of-order response would look like an OOB update which would trigger an object invalidation within the MDE **712**. However, by enforcing the order of responses, the MDE **712** can detect out-of-band writes while providing for parallel, non-overlapping, file writes. To this end, the MDE **712** enforces order by tagging each outgoing write request to the filer with an identifier which is monotonically incremented in value (e.g., a sequence number) with each write request sent. As a result, responses received by the MDE **712** can be sorted according to the sequence number value before being processed. Although such requests are sequenced and ordered, the network transport could reorder them (e.g., UDP), or the filer itself could choose to reorder their execution. In this particular case, the MDE **712** would perceive the effects of such re-ordering as an OOB update, and invalidate the file.

[0068] While a variety of different architectures may be used to implement the NAS cache appliance **710**, variations provide a hardware implementation that includes a network switch interconnect component for routing network traffic, a network processor component for packet processing, a cache controller, and cache memory component for storing cached data files. The high-speed network switch provides client and filer interfaces and multiple high-speed (e.g., 10 Gbps) connections to the packet processing and cache controller hardware. The high-speed network switch manages data flow between the client/filer I/O ports and the packet processing and cache controller hardware. The high-speed network switch may be optimized for network traffic where it is desirable to obtain extremely low latency. In addition, one or more network processor units (NPUs) are included to run the core software on the device to perform node management, packet processing, cache management, and client/filer communications. Still further, a substantial cache memory is provided for storing data files, along with a cache controller that is responsible for connecting cache memory to the high-speed network switch.

[0069] FIG. **8** illustrates a request state diagram **800** for handling out-of-band update requests at a cache node appliance. In an initial or start state (**801**), a cache node appliance **710** is positioned between the storage clients and the NAS filers, where it operates to intercept requests between the clients and filers and provide read and write cache acceleration by storing and recalling frequently used information. In this state, the metadata, or attribute, of a particular file is absent from the MDE's cache. After receiving an incoming NAS request from a client (transition **803**), the cache appliance **710** inspects the packet information associated with the request to obtain information for moving the packet through the system (e.g., network protocol traffic state parameters). The MDE determines that the attribute is missing from the cache. Consequently, it sends the request to the target filer. In the transition **803**, the MDE receives the attribute from the filer for the particular file and loads it into its cache.

[0070] The cache node appliance **710** next moves the metadata to a validation state **805** where the MDE applies a lease time on the cached metadata. In this valid state **805** all requests for metadata are satisfied from the MDE's cache **806**. At some point in time, the lease on the cached metadata expires. The next incoming request **807** detects that the lease has expired and moves the attribute to an expired state (state **809**). In addition, the MDE forwards the incoming request to the filer (transition **811**), at which point the cache node appliance **710** moves the metadata to a pending state (state **813**) where the MDE waits for a reply from the filer. If no OOB update is detected in the reply to the request (transition **814**), the cache node appliance moves the metadata to the valid state (state **805**). However, if an OOB update is detected from a reply attribute (transition **815**), the cache node appliance moves the metadata to the invalid state (state **817**). This detection process can be implemented at the cache node appliance by comparing a reply attribute from the filer with a

corresponding cached version of the metadata. If a difference is detected, then the object is invalidated (transition **819**), and the sequence returns to the start state (state **810**) to await the next incoming request.

[0071] Although illustrative embodiments have been described in detail herein with reference to the accompanying drawings, variations to specific embodiments and details are encompassed by this disclosure. It is intended that the scope of embodiments described herein be defined by claims and their equivalents. Furthermore, it is contemplated that a particular feature described, either individually or as part of an embodiment, can be combined with other individually described features, or parts of other embodiments. Thus, absence of describing combinations should not preclude the inventor(s) from claiming rights to such combinations.

What is claimed is:

1. A method for operating a cache appliance system, the method being implemented by one or more processors and comprising:

connecting to a networked file system so that the cache appliance system intercepts traffic as between a plurality of clients and the networked file system while storing data corresponding to file system objects provided by the networked file system;

detecting, at the cache appliance system, a change to the networked file system that is not a result of the intercepted traffic; and

in response to detecting the change to the networked file system, performing one or more operations to maintain coherency as between the networked file system and the stored data of the cache appliance system.

2. The method of claim **1**, wherein performing the one or more operations includes flushing the cache appliance system of at least data that corresponds to a portion of the networked file system that was detected as being changed.

3. The method of claim **1**, wherein detecting the change to the networked file system includes detecting a divergence as between metadata of the networked file system and metadata identified from stored data of the cache appliance system.

4. The method of claim **1**, wherein detecting the change to the networked file system includes analyzing metadata of file system objects provided with the networked file system, as compared to metadata of corresponding file system objects stored with the cache appliance system.

5. The method of claim **1**, wherein detecting the change to the networked file system includes assigning a shelf-life of a pre-determined duration to a file system object provided by data stored with the cache appliance system, then probing the networked file system to determine if any changes occurred to the file system object that are not reflected in the file system object stored with the cache appliance system.

6. The method of claim **5**, wherein performing one or more operations includes flushing the cache appliance system of at least the file system object.

7. The method of claim **1**, wherein detecting the change in the networked file system includes inspecting attributes provided in traffic that corresponds to replies from the networked file system.

8. The method of claim **7**, wherein inspecting attributes includes identifying an operation that is performed on the networked file system in providing a reply, then identifying at least one of a pre-operation attribute or post-operation attribute specified in the reply, and comparing the identified attribute to metadata provided with data stored in the cache appliance system for a corresponding file system object.

9. The method of claim **7**, wherein inspecting attributes includes identifying a read-type operation that is performed on the networked file system in providing a reply, then comparing a post-operation attribute specified in the reply with a corresponding metadata provided with data stored in the cache appliance system for a corresponding file system object.

10. The method of claim **7**, wherein inspecting attributes includes identifying an update-type operation that is performed on the networked file system in providing a reply, then comparing a pre-operation attribute specified in the reply with a corresponding metadata provided with data stored in the cache appliance system for a corresponding file system object.

11. The method of claim **1**, wherein detecting the change in the networked file system includes:

tagging individual outgoing write requests to the networked file system with a monotonically incremented identifier that reflects a sequence in which each write request is forwarded from the cache appliance system to the networked file system;

sorting responses from the networked file system based on the monotonically incremented identifier; and

determining that that the responses are re-ordered with respect to the outgoing write requests.

12. A computer-readable medium for operating a cache appliance system, the computer-readable medium including instructions that, when executed by one or more processors, cause the one or more processors to perform operations that comprise:

connecting to a networked file system so that the cache appliance system intercepts traffic as between a plurality of clients and the networked file system while storing data corresponding to file system objects provided by the networked file system;

detecting, at the cache appliance system, a change to the networked file system that is not a result of the intercepted traffic; and

in response to detecting the change to the networked file system, performing one or more operations to maintain coherency as between the networked file system and the stored data of the cache appliance system.

13. The computer-readable medium of claim **12**, wherein instructions for performing the one or more operations includes instructions for flushing the cache appliance system of at least data that corresponds to a portion of the networked file system that was detected as being changed.

14. The computer-readable medium of claim **12**, wherein instructions for detecting the change to the networked file system includes instructions for detecting a divergence as between metadata of the networked file system and metadata identified from stored data of the cache appliance system.

15. The computer-readable medium of claim **12**, wherein instructions for detecting the change to the networked file system includes instructions for analyzing metadata of file system objects provided with the networked file system, as compared to metadata of corresponding file system objects stored with the cache appliance system.

16. The computer-readable medium of claim **12**, wherein instructions for detecting the change to the networked file system includes instructions for assigning a shelf-life of a pre-determined duration to a file system object provided by

data stored with the cache appliance system, and instructions for then probing the networked file system to determine if any changes occurred to the file system object that are not reflected in the file system object stored with the cache appliance system.

17. The computer-readable medium of claim **12**, wherein instructions for performing one or more operations includes instructions for flushing the cache appliance system of at least the file system object.

18. The computer-readable medium of claim **12**, wherein instructions for detecting the change in the networked file system includes instructions for inspecting attributes provided in traffic that corresponds to replies from the networked file system.

19. The computer-readable medium of claim **18**, wherein instructions for inspecting attributes includes instructions for identifying an operation that is performed on the networked file system in providing a reply, and instructions for then identifying at least one of a pre-operation attribute or post-operation attribute specified in the reply, and instructions for comparing the identified attribute to metadata provided with data stored in the cache appliance system for a corresponding file system object.

20. The computer-readable medium of claim **18**, wherein instructions for inspecting attributes includes instructions for identifying a read-type operation that is performed on the networked file system in providing a reply, and instructions for then comparing a post-operation attribute specified in the reply with a corresponding metadata provided with data stored in the cache appliance system for a corresponding file system object.

21. The computer-readable medium of claim **18**, wherein instructions for inspecting attributes includes instructions for identifying an update-type operation that is performed on the networked file system in providing a reply, then comparing a pre-operation attribute specified in the reply with a corresponding metadata provided with data stored in the cache appliance system for a corresponding file system object.

* * * * *