US 20030154412A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2003/0154412 A1**

Hetzler et al. (43) **Pub. Date: Aug. 14, 2003**

(54) **SYSTEM AND METHOD FOR AUTHENTICATING BLOCK LEVEL CACHE ACCESS ON NETWORK**

(75) Inventors: **Steven R. Hetzler**, Los Altos, CA (US); **Daniel Felix Smith**, San Jose, CA (US)

Correspondence Address:
John L. Rogitz
Rogitz & Associates
Suite 3120
750 B Street
San Diego, CA 92101 (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US) (US)

(21) Appl. No.: **10/076,732**

(22) Filed: **Feb. 12, 2002**

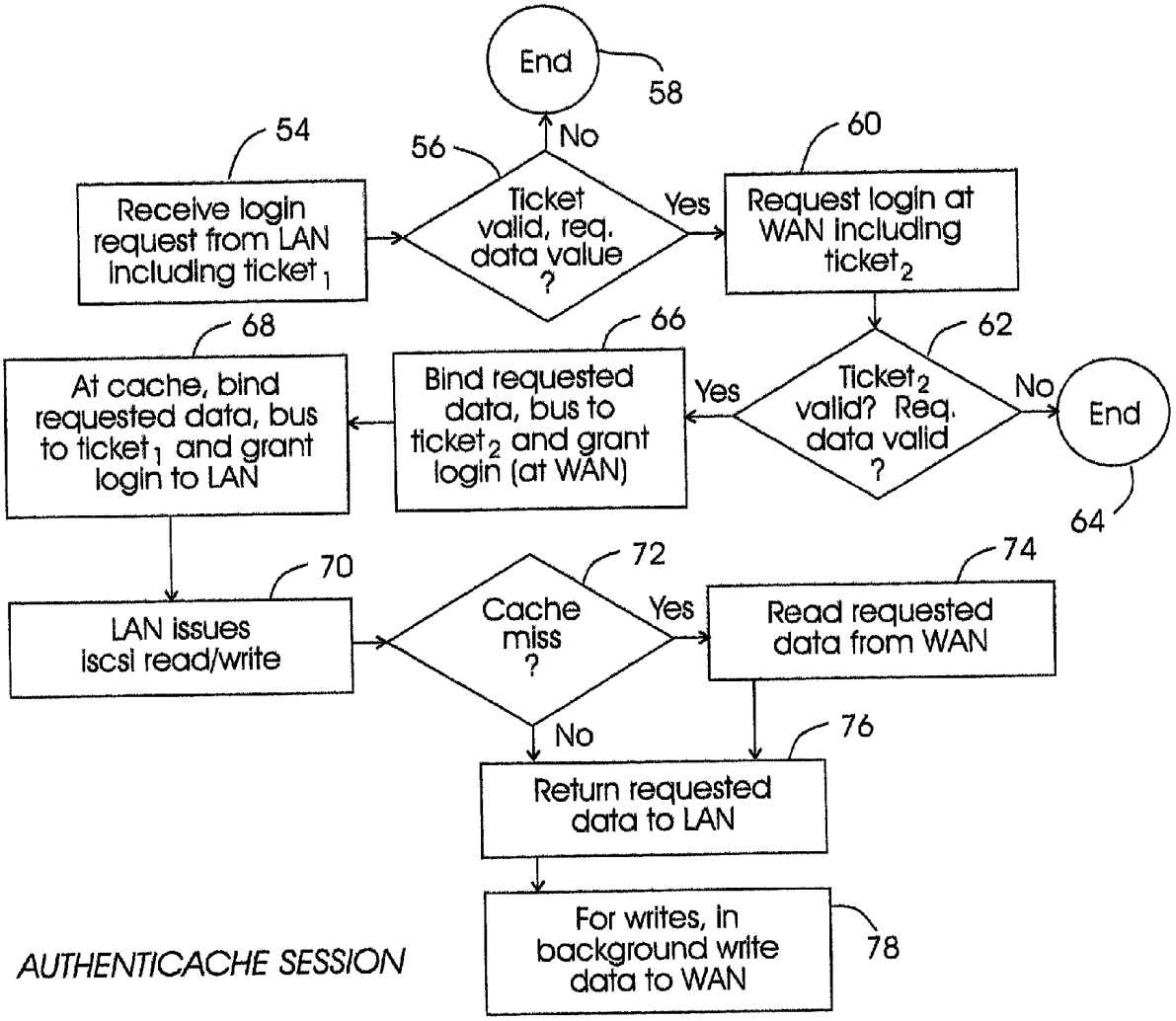
Publication Classification

(51) **Int. Cl.**[7] **H04L 9/00**

(52) **U.S. Cl.** **713/202**; 713/153

(57) **ABSTRACT**

A data cache for an iSCSI network caches block-level data from WAN servers for use by clients (e.g., LANs). The cache authenticates itself to the WAN servers, and authenticates clients requesting cache access. Mechanisms are provided to prevent clients from accessing cached data intended for other clients.



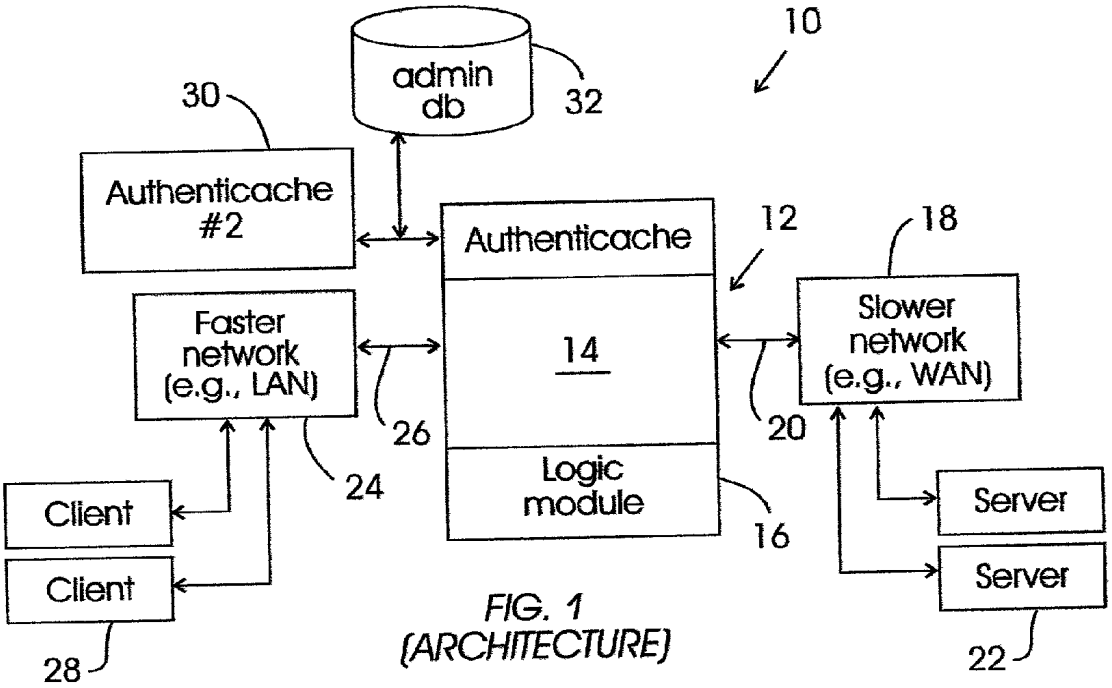


FIG. 1
(ARCHITECTURE)

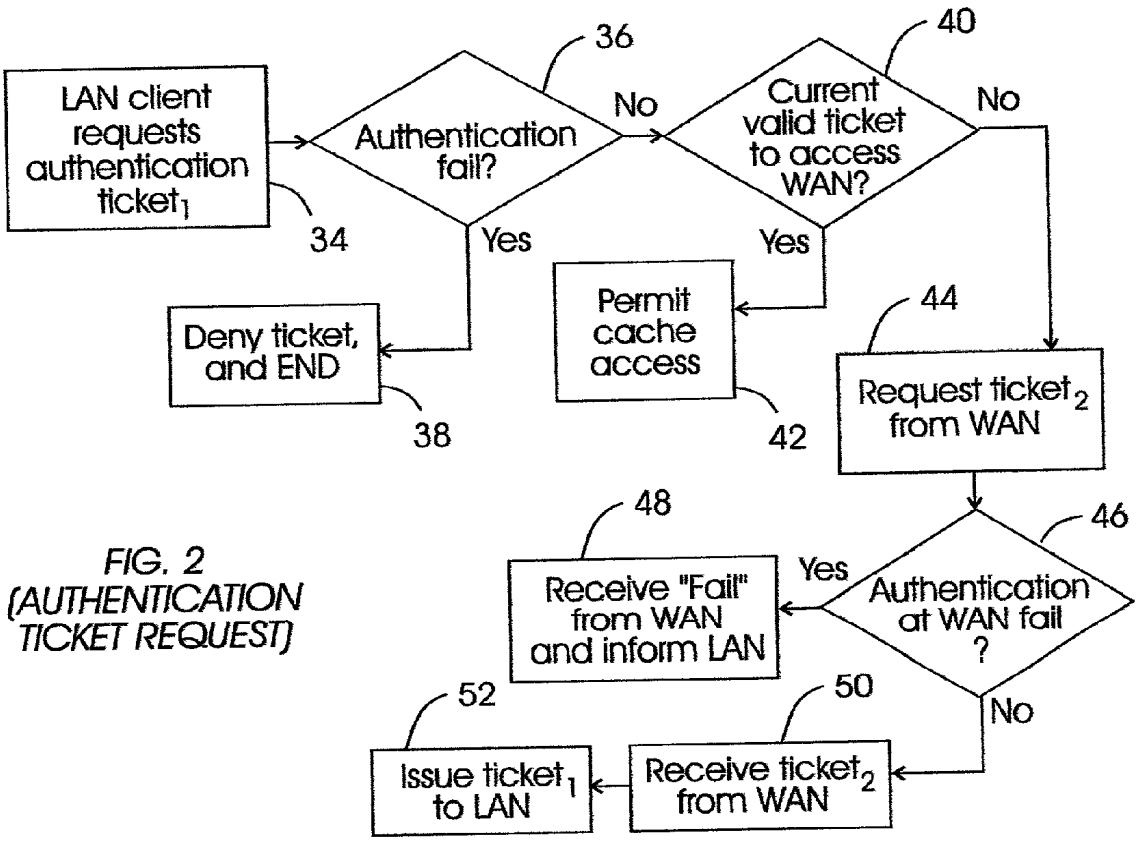


FIG. 2
(AUTHENTICATION TICKET REQUEST)

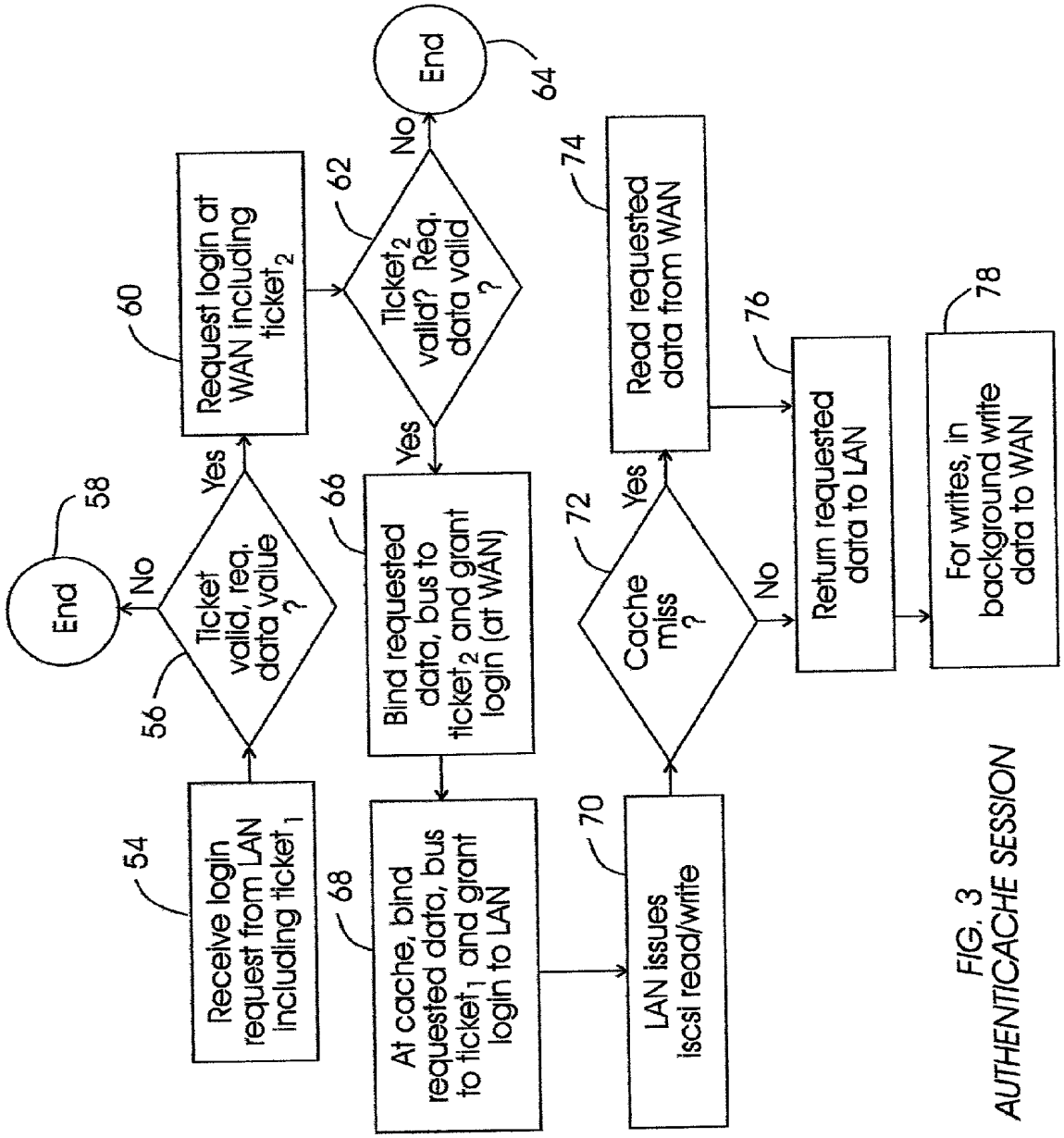


FIG. 3
AUTHENTICACHE SESSION

SYSTEM AND METHOD FOR AUTHENTICATING BLOCK LEVEL CACHE ACCESS ON NETWORK

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates to block level cache access in network environments.

[0003] 2. Description of the Related Art

[0004] One development arising from the emergence of networked computing such as the Internet is that an entity's data storage requirements can be met by data storage facilities that are remote from the entity's computers but that are accessible over the network. To provide for fast data transfer through the network, data transfer standards such as the so-called iSCSI (Internet Small Computer Systems Interface) standard have been proposed.

[0005] The iSCSI protocol is a block level protocol that is very fast and that is optimized for small, simple commands for facilitating data transfers. It is an implementation of the SCSI protocol that was originally developed to support data transfer between a processor and a local data storage, in particular a local hard disk drive. The iSCSI protocol, on the other hand, is designed to work over TCP/IP (Internet Protocol), which is a general purpose, Internet-related protocol for transferring data through a local area network (LAN) or wide area network (WAN) such as the Internet or local area network. iSCSI thus combines the advantages of peripheral communication ability of SCSI with the data transports developed for IP.

[0006] Using what was originally a local hard drive transfer protocol (SCSI) in a WAN environment, such as the Internet, poses challenges. Such protocols emphasize speed but not security, since they were developed for non-networked environments. For example, a local client computer or network (e.g., a LAN) might support a high data transfer rate and low latency, whereas a WAN holding a data repository sought to be accessed by the LAN might not support as high a rate and/or have greater latency.

[0007] As recognized herein, a cache can be used to address the above-noted problem, since a cache is a device that temporarily holds data in a fast store and thus can be used as a buffer between a fast network and a slow one. A common function of caches is speculatively pre-fetching read data in anticipation of client requests. As also recognized herein, however, current block-level cache devices do not account for data security. Instead, in block level caches a closed, secure network is assumed, or it is assumed that in an open network data security is simply not a concern. The present invention, however, understands that in the type of network-based data storage applications made possible by modern WAN protocols and in particular made possible by iSCSI, data security at a block level cache is indeed a concern, particularly since several clients might use the same cache as a buffer between their faster, local networks and a slower network such as the Internet.

[0008] The present invention has recognized the above-noted problems and provides solutions to one or more of them as disclosed below.

SUMMARY OF THE INVENTION

[0009] A general purpose computer including a cache device processor is programmed according to the inventive steps herein. The invention can also be embodied as an article of manufacture—a machine component—that is used by a digital processing apparatus and which tangibly embodies a program of instructions that are executable by the digital processing apparatus to execute the present logic. This invention is realized in a critical machine component that causes a digital processing apparatus to perform the inventive method steps herein.

[0010] Accordingly, a cache device is provided that includes a local data storage for fulfilling requests for block storage from clients to block storage servers. The device includes logic including maintaining, in the local data storage, ephemeral copies of write blocks written from write clients to the server. The logic further includes subsequently writing the write block to the server from the local data storage. Moreover, the logic includes receiving requests from read clients to read blocks from the server, requesting the read block from the server if necessary, perhaps as part of a speculative pre-fetch, and storing the blocks in the local data storage, such that subsequent requests for the blocks can be satisfied from the local storage. The device authenticates clients, prior to permitting them to access data in the local data storage.

[0011] In a preferred implementation, the server can be a WAN server, and data is transferred to and from the cache device using iSCSI protocol. The authentication can be done by associating a client authenticator, such as a client ticket, with logical units in the storage. The client ticket is selectively granted based on whether the client is successfully authenticated.

[0012] As contemplated in one preferred, non-limiting embodiment, the logic executed by the cache device also includes attempting to gain a second authentication ticket from the server so that the cache device can gain access to original data at the server. The second ticket, if granted, is associated with the original data. With the above-mentioned authentication, storage blocks effectively are locked transparently to the clients to facilitate sharing of the data among the clients.

[0013] In further non-limiting enhancements, the cache device can be in logical series with a second cache device to effectively increase the local storage size. Or, the devices can operate in parallel to share the workload. Preferably, the cache device establishes an asymmetric queue depth device, such that a number of outstanding requests to the server can exceed a number of outstanding requests from clients.

[0014] In further enhancements, the cache device may communicate with clients and servers over a network, and automatically discover new clients and servers on the network. The number of write requests can also be limited to a set of clients, either in data volume or rate of requests, for storage quota management. If desired, sets of administrative specifications can be changed by a server and at least one authenticated client.

[0015] Still further, a ticket can be automatically reapplied for if the ticket has expired or will soon expire. Encryption can be used if desired over the network. Also, the client tickets can be stored in the cache device's local storage for efficiency, with the local storage also being potentially encrypted.

[0016] In another aspect, a method for communicating data in a network between at least one client and at least one

server includes placing data at the block level onto the network, and selectively caching at least some data at a cache device located between a client and a server. The cache device has a local data storage. Using the cache device, clients can be authenticated prior permitting them to access data at the cache device.

[0017] In still another aspect, a computer program device includes a computer program storage device that is readable by a digital processing apparatus. A program is on the program storage device, and the program includes instructions that can be executed by the digital processing apparatus for caching data flowing between a server and a client in a network. The program includes means for caching data at a block level in a data storage, as well as means for authenticating a client requesting access to data in the data storage.

[0018] In yet another aspect, a system for transferring data between a relatively slower server side and a relatively faster client side over a network using iSCSI includes a data cache between the server side and client side. The cache caches data in a local storage. Preferably, for reliability the cache uses principles of redundancy for storage. Moreover, the data cache authenticates clients prior to sending data in the local storage to the clients.

[0019] The details of the present invention, both as to its structure and operation, can best be understood in reference to the accompanying drawings, in which like reference numerals refer to like parts, and in which:

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] **FIG. 1** is a schematic diagram showing the system of the present invention;

[0021] **FIG. 2** is a flow chart of the authentication ticket request logic; and

[0022] **FIG. 3** is a flow chart of the authentication session logic.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0023] Referring initially to **FIG. 1, a** system is shown, generally designated **10**, for permitting network access to a cache device, generally designated **12**, to obtain block-level data. The cache device **12** includes a local data storage **14** of, e.g., five hundred gigabytes, encrypted if desired in accordance with encryption principles known in the art, with the device **12** including a processor that executes a logic module **16** for undertaking the present invention.

[0024] In one intended embodiment, the processor of the cache device **12** is any appropriate processing device. For instance, it could be a personal computer made by International Business Machines Corporation (IBM) of Armonk, N.Y., or it can be any other appropriate computing device such as a mainframe or laptop computer or network server computer. Preferably, for reliability the cache uses principles of redundancy for storage.

[0025] In any case, the cache device **12** communicates with a relatively slower network **18** such as a wide area network, e.g., the Internet, via a network connection **20**. In turn, the network **18** communicates with one or more servers **22** that can be data repositories.

[0026] Furthermore, the cache device **12** communicates via a network connection **26** with a relatively faster network **24** such as a local area network. Network connections shown in **FIG. 1** are illustrative, and other configurations are contemplated, such as where connections **20** and **26** share common paths.

[0027] In turn, the network **24** communicates with one or more clients **28** that can be computers, systems, or other client devices requiring access to data in the repositories of the servers **22**. Accordingly, not only the clients **28** but the faster network **24** can be thought of as "clients", while not only the servers **22** but the slower network **18** can be thought of as "servers". Furthermore, while the above disclosure refers to a "slower" network **18** and a "faster" network **24**, this speed relationship can be reversed, or both networks **18**, **24** can operate at the same speed.

[0028] In any case, data can be transmitted between the networks **18**, **24** via the cache device **12** at the block level. Preferably, the protocol used on the networks **18**, **24** includes iSCSI.

[0029] With the above overview of the present architecture in mind, it is to be understood that the present logic is executed on the architecture shown in **FIG. 1** in accordance with the flow charts discussed below. The flow charts herein illustrate the structure of the logic of the present invention as embodied in computer program software. Those skilled in the art will appreciate that the flow charts illustrate the structures of logic elements, such as computer program code elements or electronic logic circuits, that function according to this invention. Manifestly, the invention is practiced in its essential embodiment by a machine component that renders the logic elements in a form that instructs a digital processing apparatus (that is, a computer) to perform a sequence of function steps corresponding to those shown.

[0030] In other words, the logic may be embodied by a computer program that is executed by a processor as a series of computer-executable instructions. These instructions may reside, for example, in RAM or on a hard drive or optical drive, magnetic tape, electronic read-only memory, or other appropriate data storage device.

[0031] If desired, a second cache device **30** that is substantially identical in configuration and operation to the cache device **12** can be provided. The devices **12**, **30** can operate in logical series with each other, increasing the effective size of the local cache storage. Moreover, an administrative database **32** can be provided that is accessible to the cache device **12**, **30** such that the cache devices **12**, **30** can operate in parallel with each other to split the caching and authentication workload. Preferably, the cache device **12** establishes an asymmetric queue depth device, such that a number of outstanding requests to a server **22** can exceed a number of outstanding requests from clients **28**.

[0032] Now referring to **FIG. 2**, the logic for requesting and granting a cache authentication ticket can be seen. It is to be understood that while the present discussion uses the term "ticket", what is broadly meant is any type of authenticator such as a token or other string in accordance with authentication principles. It is to be further understood that data, including requests, can be encrypted and decrypted as appropriate in and among the cache device **12** and networks **18**, **24**.

[0033] It is to be further understood that the cache device **12** can authenticate on behalf of, e.g., a LAN client to a WAN server, as assumed in the discussion below, or it can authenticate itself as a proxy for an (unnamed) LAN client whose identity need not be known to the WAN server. Moreover, allocation of logical storage units in the cache device **12** can be undertaken by the WAN server (when, e.g., the cache device **12** authenticates with the WAN server using the LAN client's identity) or by the cache device **12** (when, e.g., the cache device **12** authenticates itself with the WAN server as a proxy on behalf of a LAN client).

[0034] Commencing at block **34**, a LAN client requests an authentication ticket, denoted client $ticket_1$, by sending a ticket request to the cache device **12**. The ticket request includes a request to access (B, LU . . .), wherein "B" represents the relevant bus (e.g., an iSCSI bus) and the "LU"s represent logical units in the data storage **14** that can be associated with the requesting client or with the sought-after blocks. At decision diamond **36** the device **12** determines whether authentication failed, i.e., whether the LAN client is not authorized access to the cache data storage **14**. If authentication fails, the ticket request is denied and the process ends at block **38**.

[0035] On the other hand, if the requesting client is authenticated, the data cache device **12** must in turn authenticate itself with the server **22** from which the requested data originates. Accordingly, the logic flows to decision diamond **40** to determine whether the cache device **12** has a current valid authentication ticket with respect to the requested data. If so, the logic simply ends at state **40**, wherein the requesting client is permitted access to the requested data.

[0036] Otherwise, the cache device **12** must obtain a current cache authentication ticket, denoted $ticket_2$, and in that case the logic would flow to block **44**, wherein the cache device **12** requests the cache $ticket_2$ from the WAN server holding the requested data. If authentication at the WAN server fails as determined at decision diamond **46**, the logic flows to block **48** to return "fail" and deny access. If this happens, the cache device **12** informs the LAN client. If authentication is successful, however, the logic flows from decision diamond **46** to block **50**, wherein the WAN server grants the cache $ticket_2$ to the cache device **12**. Then, the cache device **12** issues a corresponding client $ticket_1$ to the requesting client at block **52**. The client $ticket_1$ can be stored in the local storage **14** if desired.

[0037] Once the above-described process has resulted in issued tickets, the logic of **FIG. 3** can be invoked to access data. Commencing at block **54**, a login request is received from a requesting client. The request includes a valid client $ticket_1$ along with (B, LU . . .). At decision diamond **56** it is determined by the cache device **12** whether (B, LU, $ticket_1$) are all valid, and if not, the logic ends at state **58**. Otherwise, the logic proceeds to block **60**.

[0038] At block **60**, the cache device **12** requests login at the WAN server including (B', LU' . . . , $ticket_2$), wherein B' is the relevant iSCSI bus at the WAN server and LU' . . . . are the logical units at the server, either associated with the cache device **12**, or holding the original data having copies stored in the local storage **14** at (LU . . .). If the server determines validity of the login request at decision diamond **62**, the logic proceeds to block **66**. An invalid login attempt ends the process at state **64**.

[0039] At block **66** the requested data (B', LU' . . .) are bound to the cache device's $ticket_2$, and login is granted. The cache device **12** then binds (B, LU . . .) to the client $ticket_1$ at block **68** and grants login to the LAN client.

[0040] The logic commencing at block **70** is then repeated as necessary for read/write requests. At block **70**, a read or write request is received at the cache device **12** from a LAN client. The request includes (B, LU, block), wherein "block" is the specific data block or blocks sought to be read (or written).

[0041] At decision diamond **72** the cache device **12** determines whether the request can be satisfied from cache, i.e., from local storage **14**. If it cannot be satisfied from local storage **14**, the logic moves to block **74** to read the requested data (B', LU', block) from the WAN server holding the original data. From block **74**, or from decision diamond **72** when the request is satisfied locally, the logic moves to block **76** wherein the cache device **12** returns the requested data (B, LU, block) to the requesting LAN client. For write operations, the logic can continue to block **78** to write the data received from the client to the WAN server in the background, on a not-to-interfere basis with other cache operations. The server then writes the block to its own repository.

[0042] It may now be appreciated that by binding authentication tickets to logical units, one client cannot read data intended for another client. Likewise, the data cache device **12** cannot grant access to data from a server unless the server has approved.

[0043] Other security methods can be used, however, in addition to or in lieu of the above. For instance, the data cache can be flushed at the end of each operation. This can be somewhat cumbersome and problematic, however. Further, the cache device **12** can be disconnected from the network and "scrubbed" after each operation, but this too can be cumbersome and can adversely impact performance. Also, encryption of data can be used, with only authorized clients being able to read encrypted data. However, if encrypted data is accessed the encryption might eventually be broken and the data read.

[0044] If desired, additional functionality can be provided. For example, the cache device **12** can automatically discover new clients **28** and servers **22** on the network. Moreover, the cache device **12** can limit the number of write requests to a set of clients **28**, either in data volume or rate of requests, to essentially establish a storage quota management system. Further, sets of administrative specifications can be changed by authenticated servers **22** and authenticated clients **28**. If desired, storage blocks in the local storage **14** can be locked transparently to the clients **28** by the above-described mechanism of associating tickets with logical units, to facilitate sharing of the cache **12** among the clients **28**. Still further, the tickets can have expirations, and a ticket that has expired or will soon expire can be automatically reapplied for based on an alert generated by the cache **12** or the granting server **22**. In one non-limiting example, the network connections **20**, **26** can use encryption. The data cache device **12** can, in accordance with cache principles known in the art, anticipate client requests based on historical requests and adapt accordingly, e.g., by appropriately placing related data close together in the storage **14**, to thereby improve client performance.

[0045] While the particular SYSTEM AND METHOD FOR AUTHENTICATING BLOCK LEVEL CACHE ACCESS ON NETWORK as herein shown and described in detail is fully capable of attaining the above-described objects of the invention, it is to be understood that it is the presently preferred embodiment of the present invention and is thus representative of the subject matter which is broadly contemplated by the present invention, that the scope of the present invention fully encompasses other embodiments which may become obvious to those skilled in the art, and that the scope of the present invention is accordingly to be limited by nothing other than the appended claims, in which reference to an element in the singular means "at least one". All structural and functional equivalents to the elements of the above-described preferred embodiment that are known or later come to be known to those of ordinary skill in the art are expressly incorporated herein by reference and are intended to be encompassed by the present claims. Moreover, it is not necessary for a device or method to address each and every problem sought to be solved by the present invention, for it to be encompassed by the present claims. Furthermore, no element, component, or method step in the present disclosure is intended to be dedicated to the public regardless of whether the element, component, or method step is explicitly recited in the claims. No claim element herein is to be construed under the provisions of 35 U.S.C. §112, sixth paragraph, unless the element is expressly recited using the phrase "means for".

What is claimed is:

1. A cache device including a local data storage for fulfilling requests for block storage from at least one client to at least one block storage server, the device including logic for undertaking method acts comprising:

maintaining, in the local data storage, an ephemeral copy of at least one write block written from at least a write client to the server;

subsequently writing the write block to the server from the local data storage;

authenticating at least a first client, prior to permitting access to data in the local data storage.

2. The device of claim 1, wherein at least one of: the write client, and cache device, are first clients.

3. The device of claim 1, wherein the server is a WAN server.

4. The device of claim 1, wherein data is transferred to and from the device using iSCSI protocol.

5. The device of claim 1, wherein the logic undertakes the authenticating act at least in part by:

associating at least one authenticator with at least one logical unit in the storage.

6. The device of claim 5, wherein the authenticator is a first authentication ticket.

7. The device of claim 6, wherein the first ticket is selectively granted based on whether the first client is successfully authenticated.

8. The device of claim 7, wherein the first ticket is further associated with a data path carrying data from the storage to the client.

9. The device of claim 7, wherein the logic executed by the device includes attempting to gain a second authentication ticket from the server to gain access to data at the server.

10. The device of claim 9, wherein the second ticket, if granted, is associated with the data.

11. The device of claim 1, wherein the device is a first device, and the first device is in logical series with a second de vice substantially identical in configuration and operation to the firs t device.

12. The device of claim 1, wherein the device is a first device, an d the first device is in combination with an administrative database and a second device substantially identical in configuration and operation to the first device, such that the first and second devices operate in parallel.

13. The device of claim 1, wherein the device establishes an asymmetric queue depth device, such that a number of outstanding requests to the server can exceed a number of outstanding requests from clients.

14. The device of claim 1, wherein the device communicates with clients and servers over a network, and the method acts undertaken by the logic include automatically discovering new clients and servers on the network.

15. The device of claim 1, wherein the method acts undertaken by the logic include limiting a number of write requests to a set of clients, using at least one of data volume, and rate of requests.

16. The device of claim 1, wherein sets of administrative specifications can be changed by a server and/or at least one authenticated client.

17. The device of claim 1, wherein the method acts undertaken by the logic include locking storage blocks transparently to the clients to facilitate sharing of the cache among the clients.

18. The device of claim 6, wherein a ticket has an expiration, and the method acts undertaken by the logic include automatically reapplying for a ticket that has expired or will soon expire.

19. The device of claim 1, wherein at least one of: the server, and the client, include a network connection operating using encryption.

20. The device of claim 6, wherein the first ticket is stored in the device.

21. The device of claim 1, wherein the local data storage is encrypted.

22. A method for communicating data in a network between at least one client and at least one server, comprising:

placing data at block level onto the network;

selectively caching at least some data at a cache device located between a client and a server, the cache device having a local data storage; and

using the cache device to authenticate at least one client prior permitting the client to access data at the cache device.

23. The method of claim 22, wherein the server is a WAN server.

24. The method of claim 22, wherein data is transferred in the network using iSCSI protocol.

25. The method of claim 22, further comprising:

associating at least one authenticator with at least one logical unit in the storage holding at least one requested block of data.

26. The method of claim 25, wherein the authenticator is a first authentication ticket.

27. The method of claim 26, wherein the first ticket is selectively granted based on whether a client is successfully authenticated.

28. The method of claim 27, wherein the first ticket is further associated with a data path carrying data from the storage to the client.

29. The method of claim 27, further comprising attempting to gain a second authentication ticket from the server to gain access to data at the server.

30. The method of claim 29, wherein the second ticket, if granted, is associated with the data.

31. The method of claim 22, wherein the cache device is a first device, and the first device is in logical series with a second device substantially identical in configuration and operation to the first device.

32. The method of claim 22, wherein the device is a first device, and the first device is in combination with an administrative database and a second device substantially identical in configuration and operation to the first device, such that the first and second devices operate in parallel.

33. The method of claim 22, wherein the device establishes an asymmetric queue depth device, such that a number of outstanding requests to the server can exceed a number of outstanding requests from clients.

34. The method of claim 22, further comprising automatically discovering new clients and servers on the network.

35. The method of claim 22, further comprising limiting a number of write requests to a set of clients, either in data volume or rate of requests.

36. The method of claim 22, wherein sets of administrative specifications can be changed by a server and/or at least one authenticated client.

37. The method of claim 22, further comprising locking storage blocks transparently to the clients to facilitate sharing of the cache among the clients.

38. The method of claim 26, wherein a ticket has an expiration, and the method includes automatically reapplying for a ticket that has expired or will soon expire.

39. The method of claim 22, wherein at least one of: the server, and the client, include a network connection operating using encryption.

40. The method of claim 26, wherein the first ticket is stored in the cache device.

41. The method of claim 22, wherein the local data storage is encrypted.

42. A computer program device comprising:

a computer program storage device readable by a digital processing apparatus; and

a program on the program storage device and including instructions executable by the digital processing apparatus for caching data flowing between a server and a client in a network, the program comprising:

means for caching data at a block level in a data storage; and

means for authenticating a client requesting access to data in the data storage.

43. The device of claim 42, further comprising:

means for maintaining, in the data storage, an ephemeral copy of at least one write block written from at least a write client to the server; and

means for subsequently writing the write block to the server from the local data storage.

44. The device of claim 42, wherein the server is a WAN server.

45. The device of claim 42, wherein data is transferred over the network using iSCSI protocol.

46. The device of claim 42, wherein the means for authenticating further comprises:

means for associating at least one authenticator with at least one logical unit in the storage holding at least one requested block of data.

47. The device of claim 46, wherein the authenticator is a first authentication ticket.

48. The device of claim 47, wherein the first ticket is selectively granted based on whether the client is successfully authenticated.

49. The device of claim 46, wherein the first ticket is further associated with a data path carrying data from the storage to the client.

50. The device of claim 48, further comprising means for attempting to gain a second authentication ticket from the server to gain access to original data at the server.

51. The device of claim 50, wherein the second ticket, if granted, is associated with the original data.

52. The device of claim 42, wherein the computer program device is a first device, and the first device is in logical series with a second device substantially identical in configuration and operation to the first device.

53. The device of claim 42, wherein the computer program device is a first device, and the first device is in combination with an administrative database and a second device substantially identical in configuration and operation to the first device, such that the first and second devices operate in parallel.

54. The device of claim 42, wherein the means for caching establishes an asymmetric queue depth device, such that a number of outstanding requests to the server can exceed a number of outstanding requests from clients.

55. The device of claim 42, further comprising means for automatically discovering new clients and servers on the network.

56. The device of claim 43, further comprising means for limiting a number of write requests to a set of clients, either in data volume or rate of requests.

57. The device of claim 42, wherein sets of administrative specifications can be changed by a server and/or at least one authenticated client.

58. The device of claim 42, wherein the means for authenticating further comprises means for locking storage blocks transparently to the clients to facilitate sharing of the cache among the clients.

59. The device of claim 47, wherein a ticket has an expiration, and the device further includes means for automatically reapplying for a ticket that has expired or will soon expire.

60. The device of claim 42, wherein at least one of: the server, and the client, include a network connection operating using encryption.

61. The device of claim 47, wherein the ticket is stored in the device.

62. A system for transferring data between a server side and a client side over a network using iSCSI, comprising:

a data cache between the server side and client side and caching data in a local storage,

the data cache authenticating a client prior to sending data in the local storage to the client.

63. The system of claim 62, wherein the data cached in local storage is at the block level.

64. The system of claim 63, wherein the server side communicates with a WAN server.

65. The system of claim 64, wherein a client communicating with the client side is authenticated at least in part by:

associating at least one authenticator with at least one logical unit in the storage holding at least one requested block of data.

66. The system of claim 65, wherein the authenticator is a first authentication ticket.

67. The system of claim 66, wherein the first ticket is selectively granted by the cache based on whether the client is successfully authenticated by the cache.

68. The system of claim 67, wherein the first ticket is further associated with a data path carrying data from the storage to the client side.

69. The system of claim 67, wherein the cache attempts to gain a second authentication ticket from a server on the server side to gain access to data at the server.

70. The system of claim 69, wherein the second ticket, if granted, is associated with the data.

71. The system of claim 70, wherein the cache serves plural clients on the client side.

72. The device of claim 5, wherein the associating act is undertaken by the server.

73. The device of claim 5, wherein the associating act is undertaken by the cache device.

74. The method of claim 25, wherein the associating act is undertaken by the server.

75. The method of claim 25, wherein the associating act is undertaken by the cache device.

76. The system of claim 65, wherein the associating act is undertaken by the server side.

77. The system of claim 65, wherein the associating act is undertaken by the cache device.

78. The device of claim 42, further comprising:

means for receiving a request from at least a read client to read at least one read block from the server; and

means for storing the read block in the local data storage, such that subsequent requests for the read block from the read client can be satisfied from the local storage.

79. The device of claim 78, wherein the means for storing the read block stores the read block as a speculative pre-fetch in anticipation of client requests.

80. The cache device of claim 1, wherein the method acts undertaken by the device include:

receiving a request from at least a read client to read at least one read block from the server;

requesting a read block from a server; and

storing the read block in the local data storage, such that subsequent requests for the read block from the read client can be satisfied from the local storage.

81. A cache device including a local data storage for fulfilling requests for block storage from at least one client to at least one block storage server, the device including logic for undertaking method acts comprising:

receiving a request from at least a read client to read at least one read block from the server;

requesting a read block from a server;

storing the read block in the local data storage, such that subsequent requests for the read block from the read client can be satisfied from the local storage; and

authenticating at least a first client, prior to permitting access to data in the local data storage.

82. The cache device of claim 81, wherein the method acts undertaken by the device include:

maintaining, in the local data storage, an ephemeral copy of at least one write block written from at least a write client to the server;

subsequently writing the write block to the server from the local data storage;

83. The device of claim 80, wherein the act of requesting a read block from a server is undertaken as a speculative pre-fetch in anticipation of client requests.

* * * * *