(54) Title: SYSTEM AND METHOD FOR EMULATING OPERATING SYSTEM METADATA TO PROVIDE CROSS-PLATFORM ACCESS TO STORAGE VOLUMES

(57) Abstract: A system and method for emulating operating system metadata to provide cross-platform access to storage volumes may include a first host and an off-host virtualizer. The off-host virtualizer may be configured to generate operating system metadata for a first virtual storage device, such as a virtual LUN, and to make the operating system metadata accessible to the first host. The metadata may be customized in accordance with a requirement of the operating system in use at the first host. The first host may include a storage software stack including a first layer, configured to use the operating system metadata to detect the existence of the first virtual storage device as an addressable storage device.
Published:
- with international search report
- before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.
TITL E: SYSTEM AND METHOD FOR EMULATING OPERATING SYSTEM METADATA TO PROVIDE CROSS-PLATFORM ACCESS TO STORAGE VOLUMES

BACKGROUND OF THE INVENTION

5 Field of the Invention

[0001] This invention relates to computer systems and, more particularly, to off-host virtualization within storage environments.

10 Description of the Related Art

[0002] Many business organizations and governmental entities rely upon applications that access large amounts of data, often exceeding a terabyte of data, for mission-critical applications. Often such data is stored on many different storage devices, which may be heterogeneous in nature, including many different types of devices from many different manufacturers.

[0003] Configuring individual applications that consume data, or application server systems that host such applications, to recognize and directly interact with each different storage device that may possibly be encountered in a heterogeneous storage environment would be increasingly difficult as the environment scaled in size and complexity. Therefore, in some storage environments, specialized storage management software and hardware may be used to provide a more uniform storage model to storage consumers. Such software and hardware may also be configured to present physical storage devices as virtual storage devices (e.g., virtual SCSI disks) to computer hosts, and to add storage features not present in individual storage devices to the storage model. For example, features to increase fault tolerance, such as data mirroring, snapshot/fixed image creation, or data parity, as well as features to increase data access performance, such as disk striping, may be implemented in the storage model via hardware or software. The added storage features may be referred to as storage virtualization features, and the software and/or hardware providing the virtual storage devices and the added storage features may be termed “virtualizers” or “virtualization controllers”. Virtualization may be performed within computer hosts, such as within a volume manager layer of a storage software stack at the host, and/or in devices external to the host, such as virtualization switches or virtualization appliances. Such external devices providing virtualization may be termed “off-host” virtualizers, and may be utilized in order to offload processing required for virtualization from the host. Off-host virtualizers may be connected to the external physical storage devices for which they provide virtualization functions via a variety of interconnects, such as Fiber Channel links, Internet Protocol (IP) networks, and the like.

[0004] Traditionally, storage software within a computer host consists of a number of layers, such as a file system layer, a disk driver layer, etc. Some of the storage software layers may form part of the operating system in use at the host, and may differ from one operating system to another. When accessing a physical disk, a layer such as the disk driver layer for a given operating system may be configured to expect certain types of configuration information for the disk to be laid out in a specific format, for example in a header (located at the first few blocks of the disk) containing disk partition layout information. The storage stack software layers used to access local physical disks may also be utilized to access external storage devices presented as virtual storage devices by off-host virtualizers. Therefore, it may be desirable for an off-host virtualizer to provide configuration information for the
virtual storage devices in formats expected by the different operating systems that may be in use at the hosts in the storage environment. In addition, it may be desirable for the off-host virtualizer to implement a technique to flexibly and dynamically map storage within external physical storage devices to the virtual storage devices presented to the host storage software layers, e.g., without requiring a reboot of the host.

SUMMARY OF THE INVENTION

[0005] Various embodiments of a system and method for emulating operating system metadata to provide cross-platform access to storage volumes are disclosed. According to a first embodiment, a system may include a first host and an off-host virtualizer, such as a virtualization switch or a virtualization appliance. The off-host virtualizer may be configured to generate operating system metadata for a first virtual storage device, such as a virtual LUN, and to make the operating system metadata accessible to the first host. The operating system metadata may be customized according to a requirement of the operating system in use at the host; that is, the off-host virtualizer may be capable of providing metadata customized for a number of different operating systems in use at different hosts. The first host may be configured to specify the specific metadata requirements of its operating system to the off-host virtualizer. The first host may include a storage software stack including a first layer, such as a disk driver layer, configured to use the operating system metadata provided by the off-host virtualizer to detect the existence of the first virtual storage device as an addressable storage device such as a partition. In another embodiment, the off-host virtualizer may be configured to aggregate storage within one or more physical storage devices into a logical volume, and to map the logical volume to the first virtual storage device or virtual LUN. In such an embodiment, the off-host virtualizer may also be configured to provide logical volume metadata to a second layer of the storage software stack (such as an intermediate driver layer between a disk driver layer and a file system layer), allowing the second layer to perform I/O operations on the logical volume.

[0006] In one embodiment, the first virtual storage device may be initially unmapped to physical storage. The recognition of the unmapped first virtual storage device as an addressable storage device may occur during a system initialization stage prior to an initiation of production I/O operations. In this way, an unmapped or "blank" virtual LUN may be prepared for subsequent dynamic mapping by the off-host virtualizer. The unmapped LUN may be given an initial size equal to the maximum allowed LUN size supported by the operating system in use at the host, so that the size of the virtual LUN may not require modification after initialization. In some embodiments, multiple virtual LUNs may be pre-generated for use at a single host, for example in order to isolate storage for different applications, or to accommodate limits on maximum LUN sizes.

[0007] In another embodiment, the system may also include two or more physical storage devices, and the off-host virtualizer may be configured to dynamically map physical storage from a first and a second physical storage device to a respective range of addresses within the first virtual storage device. For example, the off-host virtualizer may be configured to perform an N-to-1 mapping between the physical storage devices (which may be called physical LUNs) and virtual LUNs, allowing storage in the physical storage devices to be accessed from the host via the pre-generated virtual LUNs. Configuration information regarding the location of the first and/or the second address ranges within the virtual LUN (i.e., the regions of the virtual LUN that are mapped to the physical storage devices) may be passed from the off-host virtualizer to a second layer of the storage stack at the host (e.g., an intermediate driver layer above a disk driver layer) using a variety of different mechanisms. Such mechanisms may
include, for example, the off-host virtualizer writing the configuration information to certain special blocks within the virtual LUN, sending messages to the host over a network, or special extended SCSI mode pages. In one embodiment, two or more different ranges of physical storage within a single physical storage device may be mapped to corresponding pre-generated virtual storage devices such as virtual LUNs and presented to corresponding hosts. That is, the off-host virtualizer may allow each host of a plurality of hosts to access a respective portion of a physical storage device through a respective virtual LUN. In such embodiments, the off-host virtualizer may also be configured to implement a security policy isolating the ranges of physical storage within the shared physical storage device; i.e., to allow a host to access only those regions to which the host has been granted access, and to prevent unauthorized accesses.

In one embodiment, the off-host virtualizer may be further configured to aggregate storage within a physical storage device into a logical volume, dynamically map the logical volume to a range of addresses within a pre-generated virtual storage device, and pass logical volume metadata to the second layer of the storage stack, allowing I/O operations to be performed on the logical volume. Storage from a single physical storage device may be aggregated into any desired number of different logical volumes, and any desired number of logical volumes may be mapped to a single virtual storage device or virtual LUN. The off-host virtualizer may be further configured to provide volume-level security, i.e., to prevent unauthorized access from a host to a logical volume, even when the physical storage corresponding to the logical volume is part of a shared physical storage device. In addition, physical storage from any desired number of physical storage devices may be aggregated into a logical volume, thereby allowing a single volume to extend over a larger address range than the maximum allowed size of a single physical storage device. The virtual storage devices or virtual LUNs may be distributed among a number of independent front-end storage networks, such as fiber channel fabrics, and the physical storage devices backing the logical volumes may be distributed among a number of independent back-end storage networks. For example, a first host may access its virtual storage devices through a first storage network, and a second host may access its virtual storage devices through a second storage network independent from the first (that is, reconfigurations and/or failures in the first storage network may not affect the second storage network). Similarly, the off-host virtualizer may access a first physical storage device through a third storage network, and a second physical storage device through a fourth storage network. The ability of the off-host virtualizer to dynamically map storage across pre-generated virtual storage devices distributed among independent storage networks may support a robust and flexible storage environment.

**BRIEF DESCRIPTION OF THE DRAWINGS**

**[0009]** FIG. 1a is a block diagram illustrating one embodiment of a computer system.

**[0010]** FIG. 1b is a block diagram illustrating an embodiment of a system configured to utilize off-host block virtualization.

**[0011]** FIG. 2a is a block diagram illustrating the addition of operating-system specific metadata to a virtual logical unit (LUN) encapsulating a source volume, according to one embodiment.

**[0012]** FIG. 2b is a block diagram illustrating an embodiment where an off-host virtualizer is configured to include a partition table within operating system metadata.
[0013] FIG. 2c is a block diagram illustrating an embodiment where an off-host virtualizer is configured to aggregate storage within physical storage devices into a logical volume and to map the logical volume to a virtual LUN.

[0014] FIG. 3 is a block diagram illustrating an embodiment including an off-host virtualizer configured to create a plurality of virtual LUNs.

[0015] FIG. 4 is a flow diagram illustrating aspects of the operation of a host and an off-host virtualizer configured to generate operating system metadata.

[0016] FIG. 5 is a flow diagram illustrating an embodiment where an off-host virtualizer is configured to aggregate storage in one or more physical storage devices into a logical volume, and to map the logical volume to a virtual LUN.

[0017] FIG. 6 is a block diagram illustrating an example of an unmapped virtual LUN according to one embodiment.

[0018] FIG. 7 is a block diagram illustrating an embodiment including an off-host virtualizer configured to create a plurality of unmapped virtual LUNs.

[0019] FIG. 8 is a block diagram illustrating an embodiment where an off-host virtualizer is configured to dynamically map physical storage from within two different physical storage devices to a single virtual LUN.

[0020] FIG. 9 is a block diagram illustrating an embodiment where an off-host virtualizer is configured to dynamically map physical storage from within a single physical storage device to two virtual LUNs assigned to different hosts.

[0021] FIG. 10 is a block diagram illustrating an embodiment where an off-host virtualizer is configured to aggregate storage of a physical storage device into a logical volume and dynamically map the logical volume to a range of blocks of a virtual LUN.

[0022] FIG. 11 is a block diagram illustrating an embodiment where an off-host virtualizer is configured to dynamically map multiple logical volumes to a single virtual LUN.

[0023] FIG. 12 is a block diagram illustrating an embodiment where an off-host virtualizer is configured to aggregate storage from a physical storage device into two logical volumes, and to dynamically map each of the two logical volumes to a different virtual LUN.

[0024] FIG. 13 is a block diagram illustrating an embodiment employing multiple storage networks.

[0025] FIG. 14 is a block diagram illustrating an embodiment where an off-host virtualizer is configured to aggregate storage from two physical storage devices into a single logical volume.

[0026] FIG. 15 is a flow diagram illustrating aspects of the operation of a host and an off-host virtualizer according to one embodiment where the off-host virtualizer is configured to support dynamic physical LUN tunneling.

[0027] FIG. 16 is a flow diagram illustrating aspects of the operation of a host and an off-host virtualizer according to one embodiment, where off-host virtualizer 180 is configured to support dynamic volume tunneling.

[0028] FIG. 17 is a block diagram illustrating an embodiment where an off-host virtualizer comprises a virtualization switch.
FIG. 18 is a block diagram illustrating an embodiment where off-host virtualizer comprises a virtualization appliance.

FIG. 19 is a block diagram illustrating an exemplary host, according to one embodiment.

FIG. 20 is a block diagram illustrating a computer-accessible medium according to one embodiment.

While the invention is susceptible to various modifications and alternative forms, specific embodiments are shown by way of example in the drawings and are herein described in detail. It should be understood, however, that drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the invention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION

FIG. 1a is a block diagram illustrating a computer system 100 according to one embodiment. System 100 includes a host 110 coupled to a physical block device 120 via an interconnect 130. Host 110 includes a traditional software storage stack 140A that may be used to perform I/O operations on physical block device 120 via interconnect 130.

Generally speaking, a physical block device 120 may comprise any hardware entity that provides a collection of linearly addressed data blocks that can be read or written. For example, in one embodiment a physical block device may be a single disk drive configured to present all of its sectors as an indexed array of blocks. In another embodiment the physical block device may be a disk array device, or a disk configured as part of a disk array device. It is contemplated that any suitable type of storage device may be configured as a block device, such as fixed or removable magnetic media drives (e.g., hard drives, floppy or Zip-based drives), writable or read-only optical media drives (e.g., CD or DVD), tape drives, solid-state mass storage devices, or any other type of storage device. The interconnect 130 may utilize any desired storage connection technology, such as various variants of the Small Computer System Interface (SCSI) protocol, Fiber Channel, Internet Protocol (IP), Internet SCSI (iSCSI), or a combination of such storage networking technologies. The software storage stack 140A may comprise layers of software within an operating system at host 110, and may be accessed by a client application to perform I/O (input/output) on a desired physical block device 120.

In the traditional storage software stack for block device access, a client application may initiate an I/O request, for example as a request to read a block of data at a specified offset within a file. The request may be received (e.g., in the form of a read() system call) at the file system layer 112, translated into a request to read a block within a particular device object (i.e., a software entity representing a storage device), and passed to the disk driver layer 114. The disk driver layer 114 may then select the targeted physical block device 120 corresponding to the disk device object, and send a request to an address at the targeted physical block device over the interconnect 130 using the interconnect-dependent I/O driver layer 116. For example, a host bus adapter (HBA) (such as a SCSI HBA) may be used to transfer the I/O request, formatted according to the appropriate storage protocol (e.g., SCSI), to a physical link of the interconnect (e.g., a SCSI bus). At the physical block device 120, an interconnect-dependent firmware layer 122 may receive the request, perform the desired physical I/O operation at the physical storage layer 124, and send the results back to the host over the interconnect. The results (e.g., the desired blocks of the file to be
read) may then be transferred through the various layers of storage stack 140A in reverse order (i.e., from the interconnect-dependent I/O driver to the file system) before being passed to the requesting client application.

In some operating systems, a hierarchical scheme may be used for addressing storage within physical block devices 120. For example, an operating system may employ a four-level hierarchical addressing scheme of the form <"hba", "bus", "target", "lun"> for SCSI devices, including a SCSI HBA identifier ("hba"), a SCSI bus identifier ("bus"), a SCSI target identifier ("target"), and a logical unit identifier ("lun"), and may be configured to populate a device database with addresses for available SCSI devices during boot. Host 110 may include multiple SCSI HBAs, and a different SCSI adapter identifier may be used for each HBA. The SCSI adapter identifiers may be numbers issued by the operating system kernel, for example based on the physical placement of the HBA cards relative to each other (i.e., based on slot numbers used for the adapter cards). Each HBA may control one or more SCSI buses, and a unique SCSI bus number may be used to identify each SCSI bus within an HBA. During system initialization, or in response to special configuration commands, the HBA may be configured to probe each bus to identify the SCSI devices currently attached to the bus. Depending on the version of the SCSI protocol in use, the number of devices (such as disks or disk arrays) that may be attached on a SCSI bus may be limited, e.g., to 15 devices excluding the HBA itself. SCSI devices that may initiate I/O operations, such as the HBA, are termed SCSI initiators, while devices where the physical I/O may be performed are called SCSI targets. Each target on the SCSI bus may identify itself to the HBA in response to the probe. In addition, each target device may also accommodate up to a protocol-specific maximum number of "logical units" (LUNs) representing independently addressable units of physical storage within the target device, and may inform the HBA of the logical unit identifiers. A target device may contain a single LUN (e.g., a LUN may represent an entire disk or even a disk array) in some embodiments. The SCSI device configuration information, such as the target device identifiers and LUN identifiers may be passed to the disk driver layer 114 by the HBAs. When issuing an I/O request, disk driver layer 114 may utilize the hierarchical SCSI address described above.

When accessing a LUN, disk driver layer 114 may expect to see OS-specific metadata at certain specific locations within the LUN. For example, in many operating systems, the disk driver layer 114 may be responsible for implementing logical partitioning (i.e., subdividing the space within a physical disk into partitions, where each partition may be used for a smaller file system). Metadata describing the layout of a partition (e.g., a starting block offset for the partition within the LUN, and the length of a partition) may be stored in an operating-system dependent format, and in an operating system-dependent location, such as in a header or a trailer, within a LUN. In the Solaris™ operating system from Sun Microsystems, for example, a virtual table of contents (VTOC) structure may be located in the first partition of a disk volume, and a copy of the VTOC may also be located in the last two cylinders of the volume. In addition, the operating system metadata may include cylinder alignment and/or cylinder size information, as well as boot code if the volume is bootable. Operating system metadata for various versions of Microsoft Windows™ may include a "magic number" (a special number or numbers that the operating system expects to find, usually at or near the start of a disk), subdisk layout information, etc. If the disk driver layer 114 does not find the metadata in the expected location and in the expected format, the disk driver layer may not be able to perform I/O operations at the LUN.

The relatively simple traditional storage software stack 140A has been enhanced over time to help provide advanced storage features, most significantly by introducing block virtualization layers. In general, block
virtualization refers to a process of creating or aggregating logical or virtual block devices out of one or more underlying physical or logical block devices, and making the virtual block devices accessible to block device consumers for storage operations. For example, in one embodiment of block virtualization, storage within multiple physical block devices, e.g. in a fiber channel storage area network (SAN), may be aggregated and presented to a host as a single virtual storage device such as a virtual LUN (VLUN), as described below in further detail. In another embodiment, one or more layers of software may rearrange blocks from one or more block devices, such as disks, and add various kinds of functions. The resulting rearranged collection of blocks may then be presented to a storage consumer, such as an application or a file system, as one or more aggregated devices with the appearance of one or more basic disk drives. That is, the more complex structure resulting from rearranging blocks and adding functionality may be presented as if it were one or more simple arrays of blocks, or logical block devices. In some embodiments, multiple layers of virtualization may be implemented. In such embodiments, one or more block devices may be mapped into a particular virtualized block device, which may be in turn mapped into still another virtualized block device, allowing complex storage functions to be implemented with simple block devices. As used herein, the term “logical volume” refers to a virtualized block device that may be presented directly for use by a file system, database, or other applications that can directly use block devices. Further details on block virtualization, and advanced storage features supported by block virtualization, are provided below.

[0039] Block virtualization may be implemented at various places within a storage stack and the associated storage environment, in both hardware and software. For example, a block virtualization layer in the form of a volume manager, such as the VERITAS Volume Manager\textsuperscript{TM} from VERITAS Software Corporation, may be added between the disk driver layer 114 and the file system layer 112. In some storage environments, virtualization functionality may be added to host bus adapters, i.e., in a layer between the interconnect-dependent I/O driver layer 116 and interconnect 130. Block virtualization may also be performed outside the host 110, e.g., in a virtualization appliance or a virtualizing switch, which may form part of the interconnect 130. Such external devices providing block virtualization (i.e., devices that are not incorporated within host 110) may be termed off-host virtualizers or off-host virtualization controllers. In some storage environments, block virtualization functionality may be implemented by an off-host virtualizer in cooperation with a host-based virtualizer. That is, some block virtualization functionality may be performed off-host, and other block virtualization features may be implemented at the host. In another embodiment, multiple devices external to the host 110, such as two or more virtualization switches, two or more virtualization appliances, or a combination of virtualization switches and virtualization appliances, may cooperate to provide block virtualization functionality; that is, the off-host virtualizer may include multiple cooperating devices.

[0040] While additional layers may be added to the storage software stack 140A, it is generally difficult to remove or completely bypass existing storage software layers of operating systems. Therefore, off-host virtualizers may typically be implemented in a manner that allows the existing storage software layers to continue to operate, even when the storage devices being presented to the operating system are virtual rather than physical, and remote rather than local. For example, because disk driver layer 114 expects to deal with SCSI LUNs when performing I/O operations, an off-host virtualizer may present a virtualized storage device to the disk driver layer as a virtual LUN. That is, as described below in further detail, on off-host virtualizer may encapsulate, or emulate the metadata for, a LUN when providing a host 110 access to a virtualized storage device. In addition, as also described below, one or
more software modules or layers may be added to storage stack 140A to support additional forms of virtualization using virtual LUNs.

[0041] FIG. 1b is a block diagram illustrating an embodiment of system 100 configured to utilize off-host block virtualization. As shown, the system may include an off-host virtualizer 180, such as a virtualization switch or a virtualization appliance, which may be included within interconnect 130 linking host 110 to physical block device 120. Host 110 may comprise an enhanced storage software stack 140B, which may include an intermediate driver layer 113 between the disk driver layer 114 and file system layer 112. In one embodiment, off-host virtualizer 180 may be configured to map storage within physical block device 120, or multiple physical block devices 120, into a virtual storage device (e.g., a virtual LUN or VLUN), and present the virtual storage device to host 110.

[0042] The back-end physical block device 120 that is mapped to a virtual LUN may be termed a “physical LUN (PLUN)” in the subsequent description. In one embodiment, off-host virtualizer 180 may be configured to map storage from a back-end physical LUN directly to a VLUN, without any additional virtualization (i.e., without creating a logical volume). Such a technique of mapping a PLUN to a VLUN may be termed “PLUN tunneling” in the subsequent description. In another embodiment, off-host virtualizer 180 may be configured to aggregate storage within one or more physical block devices 120 as one or more logical volumes, and map the logical volumes within the address space of a VLUN presented to host 110. The terms “volume tunneling” or “logical volume tunneling” may be used herein to refer to the technique of mapping a logical volume through a VLUN. Off-host virtualizer 180 may further be configured to provide intermediate driver layer 130 with metadata or configuration information on the tunneled logical volumes, allowing intermediate driver layer 130 to locate and perform I/O operations on the logical volumes located within the virtual LUN on behalf of clients such as file system layer 112 or other applications. In some embodiments, an aliasing technique such as symbolic links may be used to associate a name with a tunneled logical volume, allowing file systems, applications and system administrators to refer to the tunneled logical volume using a conventional or recommended naming scheme. In one embodiment, the selection of a name may also be coordinated between the host 110 and the off-host virtualizer 180, e.g., using a field in an extended SCSI mode page, or through an external virtualization management infrastructure including one or more additional devices and/or hosts. File system layer 112 and applications (such as database management systems) configured to utilize intermediate driver layer 113 and lower layers of storage stack 140B may be termed “virtual storage clients” or “virtual storage consumers” herein. While off-host virtualizer 180 is shown within interconnect 130 in the embodiment depicted in FIG. 1b, it is noted that in other embodiments, off-host virtualization may also be provided within physical block device 120 (e.g., by a virtualization layer between physical storage layer 124 and firmware layer 122), or at another device outside interconnect 130. As described below in further detail, in some embodiments a number of independently configurable interconnects may be employed to link off-host virtualizer 180 to hosts 110 and back-end storage devices 120: e.g., one or more front-end interconnects 130 or storage area network (SAN) fabrics may link off-host virtualizer 180 to hosts 110, while one or more back-end interconnects 130 may link off-host virtualizer to back-end storage devices.

[0043] As described above, in some embodiments, disk driver layer 114 may expect certain operating system-specific metadata to be present at operating-system specific locations or offsets within a LUN. When presenting a virtual LUN to a host 110, therefore, in such embodiments off-host virtualizer 180 may logically insert the expected metadata at the expected locations. FIG. 2a is a block diagram illustrating the addition of operating-system specific
metadata to a virtual LUN 210 encapsulating a source volume 205, according to one embodiment. As shown, the source volume 205 consists of N blocks, numbered 0 through (N-1). The virtual LUN 210 may include two regions of inserted metadata: a header 215 containing H blocks of metadata, and a trailer 225 including T blocks of metadata. Between the header 215 and the trailer 225, blocks 220 of the virtual LUN 210 may be mapped to the source volume 205, thereby making the virtual LUN 210 a total of (H+N+T) blocks long (i.e., the virtual LUN may contain blocks numbered 0 through (H+N+T-1)). Operating-system specific metadata included in header 215 and/or trailer 225 may be used by disk driver layer 114 to recognize the virtual LUN 210 as an addressable storage device. An “addressable storage device”, as used herein, is a storage device whose blocks can be accessed (e.g., from a device driver such as a disk driver) using an address including a device identifier (such as a logical unit identifier) and an offset within the device. In some embodiments, additional configuration information or logical volume metadata may also be included within header 215 and/or trailer 225. The lengths of header 215 and trailer 225, as well as the format and content of the metadata, may vary with the operating system in use at host 110. It is noted that in some embodiments, the metadata may require only a header 215, or only a trailer 225, rather than both a header and a trailer; and in other embodiments, the metadata may be stored at any arbitrary offset within the LUN.

[0044] Off-host virtualizer 180 may be configured to customize the generated operating system metadata (e.g., in header 215 and/or trailer 225) based on the specific requirements of the operating system in use at host 110. The requirements imposed by different operating systems may differ in the type of information to be included in the metadata (e.g., whether the metadata includes a partition table), the format in which the information is maintained (e.g., the units in which a partition or volume size is specified such as kilobytes or 512-byte blocks, whether offsets are expressed in hexadecimal or decimal numerals, etc.), the location within the LUN where the metadata is to be found, etc. In some embodiments, the requirements may be sent from the host 110 to the off-host-virtualizer 180. For example, in one embodiment a virtualization management infrastructure may be implemented using one or more devices and/or additional hosts (i.e., using devices and/or hosts other than host 110 and off-host virtualizer 180). A device or host of the virtualization management infrastructure may be configured to communicate with the host 110, obtain the metadata requirements for the operating system from the host 110, and provide the requirements to the off-host virtualizer 180. In another embodiment, host 110 may be configured to provide the operating system requirements to off-host virtualizer, for example using an extension of a storage command (such as an extended SCSI request). The storage command and/or the extension may be vendor unique (i.e., a storage command may need to be extended in different ways for different hardware storage vendors) in some embodiments. In one embodiment, only an identity of the operating system in use at the host 110 (e.g., the operating system name and version number) may be provided to the off-host virtualizer 180, and the off-host virtualizer may be configured to obtain the details of the operating system requirements using the operating system identity (e.g., from a database maintained at the off-host virtualizer 180 or from an external database). Operating system identity and/or metadata requirements for a host 110 may also be specified to off-host virtualizer 180 manually in some embodiments, e.g., by a system administrator using a graphical user interface or a command-line interface. In some embodiments, a common metadata format shared by multiple operating systems may be used.

[0045] The metadata inserted within virtual LUN 210 may be stored in persistent storage, e.g., within some blocks of physical block device 120 or at storage within off-host virtualizer 180, in some embodiments, and logically concatenated with the mapped blocks 220. In other embodiments, the metadata may be maintained in non-
persistent storage (e.g., within a memory at off-host virtualizer 180) and/or generated dynamically whenever a host 110 accesses the virtual LUN 210. In some embodiments, the metadata may be generated by an external agent other than off-host virtualizer 180. The external agent may be capable of emulating metadata in a variety of formats for different operating systems, including operating systems that may not have been known when the off-host virtualizer 180 was deployed. In one embodiment, off-host virtualizer 180 may be configured to support more than one operating system; i.e., off-host virtualizer may logically insert metadata blocks corresponding to any one of a number of different operating systems when presenting virtual LUN 210 to a host 110, thereby allowing hosts with different operating systems to share access to a particular storage device 120.

As stated earlier, in some embodiments off-host virtualizer 180 may be configured to generate operating system metadata including partition layout information. FIG. 2b is a block diagram illustrating one embodiment where off-host virtualizer 180 is configured to include a partition table 255 within the generated operating system metadata header 215. The partition table 255 may provide layout or mapping information describing the offset within the VLUN 210 at which the blocks mapped to a logical partition of a physical storage device 250 may be located. In the illustrated embodiment, the layout table may contain three entries corresponding to Partition 1, Partition 2 and Partition 3 of physical storage device 250, respectively.

Off-host virtualizer may also be configured to map one or more logical volumes to the address space of a virtual LUN in some embodiments. FIG. 2c is a block diagram illustrating an embodiment where off-host virtualizer 180 is configured to aggregate storage within physical storage devices 250A and 250B into a logical volume 260, and to map logical volume 260 to blocks 261 of a virtual LUN 210. In addition to generating operating system metadata as described above, off-host virtualizer 180 may also be configured to generate metadata specific to the logical volume in such embodiments. Such logical volume metadata 263 may include information (such as a starting offset, volume length, virtualization layer information such as number of mirrors, etc.) allowing intermediate driver layer 113 to perform I/O operations on the logical volume 260. It is noted that in one embodiment, logical volume metadata may be included within header 215 or trailer 225, i.e., logical volume metadata intended for use by intermediate driver layer 113 may be co-located with operating system metadata intended for use by disk driver layer 114. While the logical volume 260 illustrated in FIG. 2c is backed by two physical storage devices 250A and 250B, in general, storage from any number of physical storage devices 250 (including from a single physical storage device) may be aggregated into a logical volume 260.

FIG. 3 is a block diagram illustrating an embodiment including an off-host virtualizer 180 configured to create a plurality of virtual LUNs (VLUNs) 210. In the illustrated embodiment, hosts 110A, 110B and 110C (collectively, hosts 110) may be coupled to off-host virtualizer 180 via interconnect 130A, and off-host virtualizer 180 may be coupled to storage devices 340A, 340B and 340C (collectively, storage devices 340) via interconnect 130B. Storage devices 340 may include physical block devices 120 as well as virtual block devices (e.g., in embodiments employing multiple layers of virtualization, as described below). Hosts 110A and 110B may utilize Operating System A, while host 110C may utilize Operating System B; that is, in general, each host among hosts 110 may support any of a number of operating systems. As shown, off-host virtualizer 180 may be configured to generate metadata for a number of VLUNs 210A-210E. VLUN 210A may be assigned to host 110A (i.e., the operating system metadata generated for VLUN 210A may be made accessible to host 110A, allowing host 110A to detect an existence of VLUN 210A as an addressable storage device). VLUN 210B may be assigned to host 110B,
while VLUNs 210C, 210D and 210E may all be assigned to host 110C. Storage within storage device 340A may be mapped to both VLUN 210A and VLUN 210E, as indicated by the dotted arrows in FIG. 3. Thus, each of two or more hosts with different operating systems (e.g., host 110A and host 110C) may be given access to the same storage device (e.g., 340A) using different VLUNs (e.g., 210A and 210E) in the illustrated embodiment. In general, off-host virtualizer 180 may be configured to support any desired total number of VLUNs 210, to assign any desired VLUNs to a given host, and to map storage from any desired combination of storage devices 340 to a given VLUN. It is noted that certain operating systems may impose a limit on the total number of LUNs (or VLUNs) that may be visible to a host 110, and/or the size of a given LUN or VLUN, and that off-host virtualizer may be configured to comply with such limits in some embodiments.

FIG. 4 is a flow diagram illustrating aspects of the operation of a host 110 and an off-host virtualizer 180 configured to generate operating system metadata as described above. Off-host virtualizer 180 may be configured to receive operating system metadata requirements (block 410) from host 110, e.g., using one of the techniques described above, such as an extended SCSI command. The off-host virtualizer 180 may then generate operating system metadata (block 420) for a virtual storage device such as a VLUN 210 in accordance with the requirements, and make the operating system metadata accessible to the host 110 (block 430). The metadata may be made accessible to host 110 via a message sent by off-host virtualizer 180 over interconnect 130A in some embodiments. In other embodiments, host 110A may be configured to send a query to off-host virtualizer 180, requesting information on the VLUNs assigned to the host 110, and the metadata may be provided in response to the query. A first layer of the storage software stack (e.g., disk driver layer 114) at host 110 may be configured to use the metadata to detect the existence of the virtual storage device as an addressable storage device (block 440). It is noted that in some embodiments, where for example all the hosts 110 in the storage environment are configured to use the same operating system, the operations corresponding to block 410 may be omitted, i.e., there may be no need to provide operating system-specific requirements to an off-host virtualizer 180 in homogeneous environments where a single operating system is in use at all the hosts 110.

FIG. 5 is a flow diagram illustrating an embodiment where an off-host virtualizer 180 is configured to aggregate storage in one or more physical storage devices into a logical volume 260, and to map the logical volume to a VLUN 210. As in FIG. 4, operating system metadata requirements may be received by off-host virtualizer 180 (block 510), and operating system metadata for a virtual storage device or VLUN 210 may be generated by off-host virtualizer in accordance with the requirements (block 520). Off-host virtualizer may be configured to aggregate storage within one or more physical storage devices 340 into a logical volume 260 (block 530) and to map the logical volume into the address space of the virtual storage device or VLUN (block 540). The operating system metadata, as well as additional metadata specific to the logical volume, may be provided to a host 110 (block 550). A first layer of a software storage stack at the host (such as disk driver layer 114) may be configured to use the operating system metadata to detect the existence of the virtual storage device (block 560), while a second layer (such as intermediate driver 113) may be configured to use the logical volume metadata to access and initiate I/O operations on the logical volume 260 (block 570).

In some embodiments, the basic techniques of PLUN tunneling and volume tunneling described above may be extended to allow dynamic association of PLUNs and/or logical volumes with VLUNs. While logical volumes such as logical volume 260 may typically be created and dynamically reconfigured (e.g., grown or shrunk,
imported to hosts 110 or exported from hosts 110) efficiently, similar configuration operations on LUNs (i.e., physical and/or virtual LUNs) may typically be fairly slow. Some LUN reconfiguration operations may be at least partially asynchronous, and may have unbounded completion times and/or ambiguous failure states. On many operating systems, LUN reconfiguration may only be completed after a system reboot; for example, a newly created physical or virtual LUN may not be detected by the operating system without a reboot. In order to be able to flexibly map logical volumes to virtual LUNs, while avoiding the delays and problems associated with LUN reconfigurations, therefore, it may be advisable to generate unmapped virtual LUNs (i.e., to create operating system metadata for virtual LUNs that are not initially mapped to any physical LUNs or to logical volumes) and pre-assign the unmapped virtual LUNs to hosts 110 as part of an initialization process. The initialization process may be completed prior to performing storage operations on the virtual LUNs on behalf of applications. During the initialization process (which may include a reboot of the system in some embodiments) the layers of the software storage stack 140B may be configured to detect the existence of the virtual LUNs as addressable storage devices. Subsequent to the initialization, off-host virtualizer 180 may dynamically map physical LUNs and/or logical volumes to the virtual LUNs (e.g., by modifying portions of the operating system metadata), as described below in further detail. The term “dynamic mapping”, as used herein, refers to a mapping of a virtual storage device (such as a VLUN) that is performed by modifying one or more blocks of metadata, and/or by communicating via one or more messages to a host 110, without requiring a reboot of the host 110 to which the virtual storage device is presented.

FIG. 6 is a block diagram illustrating an example of an unmapped virtual LUN 230 according to one embodiment. As shown, the unmapped virtual LUN 230 may include an operating system metadata header 215 and an operating system metadata trailer 225, as well as a region of unmapped blocks 235. In some embodiments, the size of the region of unmapped blocks (X blocks in the depicted example) may be set to a maximum permissible LUN or maximum logical volume size supported by an operating system, so that any subsequent mapping of a logical volume or physical LUN to the virtual LUN does not require an expansion of the size of the virtual LUN. In one alternative embodiment, the unmapped virtual LUN may consist of only the emulated metadata (e.g., header 215 and/or trailer 225), and the size of the virtual LUN may be increased when the volume or physical LUN is dynamically mapped. In such embodiments, disk driver layer 114 may have to modify some of its internal data structures when the virtual LUN is expanded, and may have to re-read the emulated metadata in order to do so. Off-host virtualizer 180 may be configured to send a metadata change notification message to disk driver layer 114 in order to trigger the re-reading of the metadata.

FIG. 7 is a block diagram illustrating an embodiment including an off-host virtualizer 180 configured to create a plurality of unmapped virtual LUNs (VLUNs) 230. As shown, more than one unmapped VLUN may be associated with a single host 110. For example, off-host virtualizer 180 may assign unmapped VLUNs 230A and 230B to host 110A, and unmapped VLUNs 230C, 230D and 230E to host 110B. In some embodiments, multiple VLUNs may be associated with a given host to allow for isolation of storage used for different applications, or to allow access to storage beyond the maximum allowable LUN size supported in the system. Off-host virtualizer 180 may be configured to dynamically map physical and/or virtual storage from storage devices 340 to the unmapped virtual LUNs 230. Hosts 110A and 110B may be configured to use different operating systems in some embodiments, and may utilize the same operating system in other embodiments. After the physical or virtual storage has been dynamically mapped, the previously unmapped VLUN 230 may provide the same general functionality as
described previously for VLUNs 210 provided in basic (i.e., non-dynamic) PLUN tunneling and basic volume tunneling. A number of different techniques for dynamic mapping of physical storage and logical volumes to VLUNs are described below. It is noted that each such technique (e.g., a mapping from two PLUNs to a single VLUN) may also be implemented using basic PLUN tunneling or basic volume tunneling in some embodiments, i.e., without starting with unmapped VLUNs pre-assigned to hosts 110.

[0054] As described earlier, operating system-specific metadata (e.g., in header 215 and/or trailer 225 of FIG. 2b) generated by off-host virtualizer 180 may allow disk driver layer 114 at a host 110 to detect the existence of an unmapped virtual LUN 230 as an addressable storage device. After VLUN 230 has been recognized by disk driver layer 114, a block at any offset within the VLUN address space may be accessed by the disk driver layer 114, and thus by any other layer above the disk driver layer. For example, intermediate driver layer 113 may be configured to communicate with off-host virtualizer 180 by reading from, and/or writing to, a designated set of blocks emulated within VLUN 230. Such designated blocks may provide a mechanism for off-host virtualizer 180 to provide intermediate driver layer 113 with configuration information associated with logical volumes or physical LUNs mapped to VLUN 230 in some embodiments.

[0055] In one embodiment, off-host virtualizer 180 may be configured to dynamically map storage from a back-end physical LUN directly to an unmapped VLUN 230, without any additional virtualization (i.e., without creating a logical volume). Such a technique of dynamically mapping a PLUN to a VLUN 230 may be termed "dynamic PLUN tunneling" (in contrast to the basic PLUN tunneling described above, where unmapped VLUNs may not be used). Each PLUN may be mapped to a corresponding VLUN 230 (i.e., a 1-to-1 mapping of PLUNs to VLUNs may be implemented by off-host virtualizer 180) in some embodiments. In other embodiments, as described below in conjunction with the description of FIG. 8, storage from multiple PLUNs may be mapped into subranges of a given VLUN 230. In general, both basic and dynamic PLUN tunneling may allow the off-host virtualizer 180 to act as an isolation layer between VLUNs 230 (the storage entities directly accessible to hosts 110) and back-end PLUNs, allowing the off-host virtualizer to hide details related to physical storage protocol implementation from the hosts. In one implementation, for example, the back-end PLUNs may implement a different version of a storage protocol (e.g., SCSI-3) than the version seen by hosts 100 (e.g., SCSI-2), and the off-host virtualizer may provide any needed translation between the two versions. In another implementation, off-host virtualizer 180 may be configured to implement a cooperative access control mechanism for the back-end PLUNs, and the details of the mechanism may remain hidden from the hosts 110.

[0056] In addition, off-host virtualizer 180 may also be configured to increase the level of data sharing using PLUN tunneling (i.e., either basic PLUN tunneling or dynamic PLUN tunneling). Disk array devices often impose limits on the total number of concurrent "logins", i.e., the total number of entities that may access a given disk array device. In a storage environment employing PLUN tunneling for disk arrays (i.e., where the PLUNs are disk array devices), off-host virtualizers 180 may allow multiple hosts to access the disk arrays through a single login. That is, for example, multiple hosts 110 may log in to the off-host virtualizer 180, while the off-host virtualizer may log in to a disk array PLUN once on behalf of the multiple hosts 110. Off-host virtualizer 180 may then pass on I/O requests from the multiple hosts 110 to the disk array PLUN using a single login. The number of logins (i.e., distinct entities logged in) as seen by a disk array PLUN may thereby be reduced as a result of PLUN tunneling, without reducing the number of hosts 110 from which I/O operations targeted at the disk array PLUN may be initiated. The total
number of hosts 110 that may access storage at a single disk array PLUN with login count restrictions may thereby be increased, thus increasing the overall level of data sharing.

[0057] FIG. 8 is a block diagram illustrating an embodiment where an off-host virtualizer 180 is configured to dynamically map physical storage from within two different physical storage devices 340A and 340B to a single VLUN 230B. That is, off-host virtualizer 180 may be configured to map a first range of physical storage from device 340A into a first region of mapped blocks 821A within VLUN 230B, and map a second range of physical storage from device 340B into a second region of mapped blocks 821B within VLUN 230B. The first and second ranges of physical storage may each represent a respective PLUN, such as a disk array, or a respective subset of a PLUN. Configuration information indicating the offsets within VLUN 230B at which mapped blocks 821A and 821B are located may be provided by off-host virtualizer 180 to intermediate driver layer 113 using a variety of mechanisms in different embodiments. For example, in one embodiment, off-host virtualizer 180 may write the configuration information to a designated set of blocks within VLUN 230, and intermediate driver layer 113 may be configured to read the designated set of blocks, as described above. In another embodiment, off-host virtualizer 180 may send a message containing the configuration information to host 110A, either directly (over interconnect 350A or another network) or through an intermediate coordination server. In yet another embodiment, the configuration information may be supplied within a special SCSI mode page (i.e., intermediate driver layer 113 may be configured to read a special SCSI mode page containing configuration information updated by off-host virtualizer 180). Combinations of these techniques may be used in some embodiments: for example, in one embodiment off-host virtualizer 180 may send a message to intermediate driver layer 113 requesting that intermediate driver layer read a special SCSI mode page containing the configuration information.

[0058] FIG. 9 is a block diagram illustrating an embodiment where off-host virtualizer 180 is configured to dynamically map physical storage from within a single physical storage device 340A to two VLUNs 230 assigned to different hosts 110A and 110B. As shown, a first range of physical storage 955A of physical storage device 340A may be mapped to a first range of mapped blocks 821A within VLUN 230B assigned to host 110A. A second range of physical storage 955B of the same physical storage device 340A may be mapped to a second range of mapped blocks 821C of VLUN 230E assigned to host 110B. In addition, in some embodiments, off-host virtualizer 180 may be configured to prevent unauthorized access to physical storage range 955A from host 110B, and to prevent unauthorized access to physical storage range 955B from host 110A. Thus, in addition to allowing access to a single physical storage device 340A from multiple hosts 110, off-host virtualizer 180 may also be configured to provide security for each range of physical storage 955A and 955B, e.g., in accordance with a specified security protocol. In one embodiment, for example, the security protocol may allow I/O operations to a given VLUN 230 (and to its backing physical storage) from only a single host 110. Off-host virtualizer 180 may be configured to maintain access rights information for the hosts 110 and VLUNs 230 in some embodiments, while in other embodiments security tokens may be provided to each host 110 indicating the specific VLUNs to which access from the host is allowed, and the security tokens may be included with I/O requests.

[0059] In some embodiments off-host virtualizer 180 may be configured to aggregate physical storage into a logical volume, and dynamically map the logical volume to an address range within a VLUN 230. Such a technique of dynamically mapping a logical volume to a VLUN 230 may be termed "dynamic volume tunneling" (in contrast to the basic volume tunneling described above, where unmapped VLUNs may not be used). FIG. 10 is a block
diagram illustrating an embodiment where off-host virtualizer 180 is configured to aggregate storage 1055A of physical storage device 340A into a logical volume 1060A, and dynamically map logical volume 1060A to a range of blocks (designated as mapped volume 1065A in FIG. 10) of VLUN 230B. As in the case of PLUN tunneling, configuration information or metadata associated with the tunnelled logical volume 1060A may be provided to intermediate driver layer 113 using any of a variety of mechanisms, such as an extended SCSI mode page, emulated virtual blocks within VLUN 230A, and/or direct or indirect messages sent from off-host virtualizer 180 to host 110A. While logical volume 1060A is shown as being backed by a portion of a single physical storage device 340A in the depicted embodiment, in other embodiments logical volume 1060A may be aggregated from all the storage within a single physical storage device, or from storage of two or more physical devices. In some embodiments employing multiple layers of virtualization, logical volume 1060A may itself be aggregated from other logical storage devices rather than directly from physical storage devices. In one embodiment, each host 110 (i.e., host 110B in addition to host 110A) may be provided access to logical volume 1060A via a separate VLUN, while in another embodiment different sets of logical volumes may be presented to different hosts 110.

FIG. 11 is a block diagram illustrating an embodiment where off-host virtualizer 180 is configured to dynamically map multiple logical volumes to a single VLUN 230. As shown, off-host virtualizer 180 may be configured to aggregate storage range 1155C from physical storage device 340A, and physical storage range 1155B from physical storage device 340B, into a logical volume 1160A, and map logical volume 1160A to a first mapped volume region 1165A of VLUN 230B. In addition, off-host virtualizer 180 may also aggregate physical storage range 1155B from physical storage device 340A into a second logical volume 1160B, and map logical volume 760B to a second mapped volume region 1165B of VLUN 230B. In general, off-host virtualizer 180 may aggregate any suitable selection of physical storage blocks from one or more physical storage devices 340 into one or more logical volumes, and map the logical volumes to one or more of the pre-generated unmapped VLUNs 230.

FIG. 12 is a block diagram illustrating another embodiment, where off-host virtualizer 180 is configured to aggregate storage from physical storage device 340A into logical volumes 1260A and 1260B, and to dynamically map each of the two logical volumes to a different VLUN 230. For example, as shown, logical volume 1260A may be mapped to a first address range within VLUN 230B, accessible from host 110A, while logical volume 1260B may be mapped to a second address range within VLUN 230E, accessible from host 110B. Off-host virtualizer 180 may further be configured to implement a security protocol to prevent unauthorized access and/or data corruption, similar to the security protocol described above for PLUN tunneling. Off-host virtualizer 180 may implement the security protocol at the logical volume level: that is, off-host virtualizer 180 may prevent unauthorized access to logical volumes 1260A (e.g., from host 110B) and 1260B (e.g., from host 110A) whose data may be stored within a single physical storage device 340A. In one embodiment, off-host virtualizer 180 may be configured to maintain access rights information for logical volumes 1260 to which each host 110 has been granted access. In other embodiments security tokens may be provided to each host 110 (e.g., by off-host virtualizer 180, or by an external security server) indicating the specific logical volumes 1260 to which access from the host is allowed, and the security tokens may be included with I/O requests.

Many storage environments utilize storage area networks (SANs), such as fibre channel fabrics, to access physical storage devices. SAN fabric reconfiguration (e.g., to provide access to a particular PLUN or logical volume from a particular host that did not previously have access to the desired PLUN or logical volume), which
may require switch reconfigurations, recabling, rebooting, etc., may typically be fairly complex and error-prone. The techniques of dynamic PLUN tunneling and dynamic volume tunneling, described above, may allow a simplification of SAN reconfiguration operations. By associating pre-generated, unmapped VLUNs to hosts, and mapping PLUNs and logical volumes to VLUNs dynamically as needed, many reconfiguration operations may require only a change of a mapping table at a switch, and a recognition of new metadata by intermediate driver layer 113. Storage devices may be more easily shared across multiple hosts 110, or logically transferred from one host to another, using PLUN tunneling and/or volume tunneling. Allocation and/or provisioning of storage, e.g., from a pool maintained by a coordinating storage allocator, may also be simplified.

[0063] In addition to simplifying SAN configuration changes, PLUN tunneling and volume tunneling may also support storage interconnection across independently configured storage networks (e.g., interconnection across multiple fiber channel fabrics). FIG. 13 is a block diagram illustrating an embodiment employing multiple storage networks. As shown, off-host virtualizer 180 may be configured to access physical storage device 340A via a first storage network 1310A, and to access physical storage device 340B via a second storage network 1310B. Off-host virtualizer 180 may aggregate storage from physical storage device 340A into logical volume 1360A, and map logical volume 1360A to VLUN 230B. Similarly, off-host virtualizer 180 may aggregate storage from physical storage device 340B into logical volume 1360B, and map logical volume 1360B to VLUN 230E. Host 110A may be configured to access VLUN 230A via a third storage network 1310C, and to access VLUN 230B via a fourth storage network 1310D.

[0064] Each storage network 1310 (i.e., storage network 1310A, 1310B, 1310C, or 1310D) may be independently configurable: that is, a reconfiguration operation performed within a given storage network 1310 may not affect any other storage network 1310. A failure or a misconfiguration within a given storage network 1310 may also not affect any other independent storage network 1310. In some embodiments, hosts 110 may include multiple HBAs, allowing each host to access multiple independent storage networks. For example, host 110A may include two HBAs in the embodiment depicted in FIG. 13, with the first HBA allowing access to storage network 1310C, and the second HBA to storage network 1310D. In such an embodiment, host 110A may be provided full connectivity to back-end physical storage devices 340, while still maintaining the advantages of configuration isolation. While FIG. 13 depicts the use of multiple independent storage networks in conjunction with volume tunneling, in other embodiments multiple independent storage networks may also be used with PLUN tunneling, or with a combination of PLUN and volume tunneling. In addition, it is noted that in some embodiments, the use of independent storage networks 1310 may be asymmetric: e.g., in one embodiment, multiple independent storage networks 1310 may be used for front-end connections (i.e., between off-host virtualizer 180 and hosts 110), while only a single storage network may be used for back-end connections (i.e., between off-host virtualizer 180 and physical storage devices 340). Any desired interconnection technology and/or protocol may be used to implement storage networks 1310, such as fiber channel, IP-based protocols, etc. For example, in one embodiment each storage network 1310 may be a fiber channel fabric. In another embodiment, the interconnect technology or protocol used within a first storage network 1310 may differ from the interconnect technology or protocol used within a second storage network 1310.

[0065] In one embodiment, volume tunneling may also allow maximum LUN size limitations to be overcome. For example, the SCSI protocol may be configured to use a 32-bit unsigned integer as a LUN block address, thereby
limiting the maximum amount of storage that can be accessed at a single LUN to 2 terabytes (for 512-byte blocks) or 32 terabytes (for 8-kilobyte blocks). Volume tunneling may allow an intermediate driver layer 113 to access storage from multiple physical LUNs as a volume mapped to a single VLUN, thereby overcoming the maximum LUN size limitation. FIG. 14 is a block diagram illustrating an embodiment where off-host virtualizer 180 may be configured to aggregate storage from two physical storage devices 340A and 340B into a single logical volume 1460A, where the size of the volume 1460A exceeds the allowed maximum LUN size supported by the storage protocol in use at storage devices 340. Off-host virtualizer 180 may further be configured to dynamically map logical volume 1460A to VLUN 230B, and to provide logical volume metadata to intermediate driver layer 113 at host 110A. The logical volume metadata may include sufficient information for intermediate driver layer 113 to access a larger address space within VLUN 230B than the maximum allowed LUN size.

[0066] FIG. 15 is a flow diagram illustrating aspects of the operation of host 110 and off-host virtualizer 180 according to one embodiment, where off-host virtualizer 180 is configured to support dynamic PLUN tunneling. After receiving operating system metadata requirements (block 1505), off-host virtualizer 180 may be configured to generate operating system metadata for an unmapped virtual storage device (e.g., a VLUN) (block 1510) and make the metadata accessible to a host 110 (block 1515). A first layer of a storage software stack at host 110, such as disk driver layer 114 of FIG. 1b, may be configured to use the O.S. metadata to detect the existence of the virtual storage device as an addressable storage device (e.g., as a LUN) (block 1520), for example during system initialization or boot. After the unmapped virtual storage device is detected, off-host virtualizer 180 may be configured to dynamically map physical storage from one or more back-end physical storage devices 340 (e.g., PLUNs) to an address range within the virtual storage device (block 1525).

[0067] FIG. 16 is a flow diagram illustrating aspects of the operation of a host 110 and an off-host virtualizer 180 according to one embodiment, where off-host virtualizer 180 is configured to support dynamic volume tunneling. The first three blocks depicted in FIG. 16 may represent functionality similar to the first three blocks shown in FIG. 15. That is, off-host virtualizer 180 may be configured to receive operating system metadata requirements (block 1605), generate operating system metadata for an unmapped virtual storage device (e.g., a VLUN) (block 1610) and make the metadata accessible to a host 110 (block 1615). A first layer of a storage software stack, such as disk driver layer 114 of FIG. 1b, may be configured to use the O.S. metadata to detect the existence of the virtual storage device as an addressable storage device (e.g., as a LUN) (block 1620), for example during system initialization or boot. In addition, off-host virtualizer 180 may be configured to aggregate storage at one or physical storage devices 340 into a logical volume (block 1625), and to dynamically map the logical volume to an address range within the previously unmapped virtual storage device (block 1630). Off-host virtualizer 180 may further be configured to provide logical volume metadata to a second layer of the storage software stack at host 110 (e.g., intermediate driver layer 113) (block 1635), allowing the second layer to locate the blocks of the logical volume and to perform desired I/O operations on the logical volume (block 1640).heer

[0068] In various embodiments, off-host virtualizer 180 may implement numerous different types of storage functions using block virtualization. For example, in one embodiment a virtual block device such as a logical volume may implement device striping, where data blocks may be distributed among multiple physical or logical block devices, and/or device spanning, in which multiple physical or logical block devices may be joined to appear as a single large logical block device. In some embodiments, virtualized block devices may provide mirroring and
other forms of redundant data storage, the ability to create a snapshot or static image of a particular block device at a point in time, and/or the ability to replicate data blocks among storage systems connected through a network such as a local area network (LAN) or a wide area network (WAN), for example. Additionally, in some embodiments virtualized block devices may implement certain performance optimizations, such as load distribution, and/or various capabilities for online reorganization of virtual device structure, such as online data migration between devices. In other embodiments, one or more block devices may be mapped into a particular virtualized block device, which may be in turn mapped into still another virtualized block device, allowing complex storage functions to be implemented with simple block devices. More than one virtualization feature, such as striping and mirroring, may thus be combined within a single virtual block device in some embodiments, creating a logically hierarchical virtual storage device.

[0069] The off-host virtualizer 180, either alone or in cooperation with one or more other virtualizers such as a volume manager at host 110 or other off-host virtualizers, may provide functions such as configuration management of virtualized block devices and distributed coordination of block device virtualization. For example, after a reconfiguration of a logical volume shared by two hosts 110 (e.g., when the logical volume is expanded, or when a new mirror is added to the logical volume), the off-host virtualizer 180 may be configured to distribute metadata or a volume description indicating the reconfiguration to the two hosts 110. In one embodiment, once the volume description has been provided to the hosts, the storage stacks at the hosts may be configured to interact directly with various storage devices 340 according to the volume description (i.e., to transform logical I/O requests into physical I/O requests using the volume description). Distribution of a virtualized block device as a volume to one or more virtual device clients, such as hosts 110, may be referred to as distributed block virtualization.

[0070] As noted previously, in some embodiments, multiple layers of virtualization may be employed, for example at the host level as well as at an off-host level, such as at a virtualization switch or at a virtualization appliance. In such embodiments, some aspects of virtualization may be visible to a virtual device consumer such as file system layer 112, while other aspects may be implemented transparently by the off-host level. Further, in some multilayer embodiments, the virtualization details of one block device (e.g., one volume) may be fully defined to a virtual device consumer (i.e., without further virtualization at an off-host level), while the virtualization details of another block device (e.g., another volume) may be partially or entirely transparent to the virtual device consumer.

[0071] In some embodiments, a virtualizer, such as off-host virtualizer 180, may be configured to distribute all defined logical volumes to each virtual device client, such as host 110, present within a system. Such embodiments may be referred to as symmetric distributed block virtualization systems. In other embodiments, specific volumes may be distributed only to respective virtual device consumers or hosts, such that at least one volume is not common to two virtual device consumers. Such embodiments may be referred to as asymmetric distributed block virtualization systems.

[0072] It is noted that off-host virtualizer 180 may be any type of device, external to host 110, that is capable of providing the virtualization functionality, including PLUN and volume tunneling, described above. For example, off-host virtualizer 180 may include a virtualization switch, a virtualization appliance, a special additional host dedicated to providing block virtualization, or an embedded system configured to use application specific integrated circuit (ASIC) or field-programmable gate array (FPGA) technology to execute provide block virtualization functionality. FIG. 17 is a block diagram illustrating an embodiment where off-host virtualizer 180 comprises a
virtualization switch 1710, and FIG. 18 is a block diagram illustrating another embodiment where off-host virtualizer 180 comprises a virtualization appliance 1810. A virtualization switch 1710 may be an intelligent fibre channel switch, configured with sufficient processing capacity to perform virtualization functions in addition to providing fibre channel connectivity. A virtualization appliance 1810 may be an intelligent device programmed to provide virtualization functions, such as mirroring, striping, snapshots, and the like. An appliance may differ from a general-purpose computer in that the appliance software is normally customized for the function (such as virtualization) to which the appliance is dedicated, pre-installed by the vendor, and not easily modifiable by a user. In some embodiments, off-host block virtualization may be provided by a collection of cooperating devices, such as two or more virtualizing switches, instead of a single device. Such a collection of cooperating devices may be configured for failover, i.e., a standby cooperating device may be configured to take over the virtualization functions supported by a failed cooperating device. An off-host virtualizer 180 may incorporate one or more processors, as well as volatile and/or non-volatile memory. In some embodiments, configuration information associated with virtualization may be maintained at a database separate from the off-host virtualizer 180, and may be accessed by off-host virtualizer over a network. In one embodiment, an off-host virtualizer may be programmable and/or configurable. Numerous other configurations of off-host virtualizer 180 are possible and contemplated.

[0073] FIG. 19 is a block diagram illustrating an exemplary host 110, according to one embodiment. Host 110 may be any computer system, such as a server comprising one or more processors 1910 and one or more memories 1920, capable of supporting the storage software stack 140B described above. As shown, host 110 may include one or more local storage devices 1930 (such as disks) as well as one or more network interfaces 1940 providing an interface to an interconnect 130. Portions of the storage software stack 140B may be resident in a memory 1920, and may be loaded into memory 1920 from storage devices 1930 as needed. In some embodiments, a host 110 may also be a diskless computer, configured to access storage from a remote location instead of using local storage devices 1930. Various other components, such as video cards, monitors, mice, etc. may also be included within a host 110 in different embodiments. Any desired operating system may be used at a host 110, including various versions of Microsoft Windows™, Solaris™ from Sun Microsystems, various versions of Linux, other operating systems based on UNIX, and the like. The intermediate driver layer 113 may be included within a volume manager in some embodiments.

[0074] FIG. 20 is a block diagram illustrating a computer-accessible medium 2000 comprising virtualization software 2010 capable of providing the functionality of off-host virtualizer 180 and block storage software stack 140B described above. Virtualization software 2010 may be provided to a computer system using a variety of computer-accessible media including electronic media (e.g., flash memory), magnetic media such as RAM (e.g., SDRAM, RDRAM, SRAM, etc.), optical storage media such as CD-ROM, etc., as well as transmission media or signals such as electrical, electromagnetic or digital signals, conveyed via a communication medium such as a network and/or a wireless link.

[0075] Although the embodiments above have been described in considerable detail, numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.
WHAT IS CLAIMED IS:

1. A system comprising:
   a first host; and
   an off-host virtualizer;
   wherein the off-host virtualizer is configured to:
   generate first operating system metadata for a first virtual storage device; and
   make the first operating system metadata accessible to the first host;
   wherein the first host comprises a storage software stack including a first layer;
   wherein the first layer is configured to use the first operating system metadata to detect an existence of the first virtual storage device as a first addressable storage device.

2. The system as recited in Claim 1, wherein the off-host virtualizer is further configured to customize the operating system metadata in response to a requirement of a particular operating system in use at the first host.

3. The system as recited in Claim 2, wherein the first host is configured to provide the requirement of the particular operating system to the off-host virtualizer.

4. The system as recited in Claim 1, further comprising a storage device, wherein the first addressable storage device is a partition of the storage device.

5. The system as recited in Claim 1, further comprising one or more physical storage devices, wherein the storage software stack includes a second layer, wherein the off-host virtualizer is further configured to:
   aggregate storage in the one or more physical storage devices into a logical volume;
   map the logical volume to an address space of the first virtual storage device;
   provide logical volume metadata for the logical volume to the second layer of the storage software stack;
   wherein the second layer of the storage software stack is configured to use the logical volume metadata to access the logical volume for I/O operations.

6. The system as recited in Claim 1, wherein the first virtual storage device is a virtual logical unit (LUN).

7. The system as recited in Claim 1, wherein the off-host virtualizer includes a virtualization switch.

8. The system as recited in Claim 1, wherein the off-host virtualizer includes a virtualization appliance.

9. The system as recited in Claim 1, wherein the first virtual storage device is unmapped to physical storage.

10. The system as recited in Claim 9, further comprising two or more physical storage devices, wherein the off-host virtualizer is further configured to dynamically map each physical storage device of the two or more physical storage devices to a respective address range within the first virtual storage device.
11. The system as recited in Claim 9, further comprising a physical storage device and a second host, wherein the off-host virtualizer is further configured to:
   generate second operating system metadata for a second virtual storage device, wherein the second virtual storage device is unmapped to physical storage; and
   make the second operating system metadata accessible to the second host;
wherein the second host comprises a second storage software stack including a first layer;
wherein the first layer of the second storage software stack is configured to use the second operating system metadata to detect an existence of the second virtual storage device as a second addressable storage device;
wherein the off-host virtualizer is further configured to:
   dynamically map a first physical address range of the physical storage device to a first virtual address range within the first virtual storage device;
   dynamically map a second physical address range of the physical storage device to a second virtual address range within the second virtual storage device; and
   prevent access to the second physical address range from the first host.

12. The system as recited in Claim 9, further comprising a first physical storage device, wherein the storage software stack includes a second layer, wherein the off-host virtualizer is further configured to:
   aggregate a first range of physical storage within the first physical storage device into a first logical volume;
   dynamically map a first range of virtual storage within the first virtual storage device to the first logical volume; and
   provide first logical volume metadata for the first logical volume to the second layer of the storage software stack;
wherein the second layer of the storage software stack is configured to use the first logical volume metadata to access the first logical volume for I/O operations.

13. The system as recited in Claim 12, wherein the off-host virtualizer is further configured to:
   aggregate a second range of physical storage within the first physical storage device into a second logical volume;
   dynamically map a second range of virtual storage within the first virtual storage device to the second logical volume; and
   provide additional logical volume metadata for the second logical volume to the second layer of the storage software stack;
wherein the second layer of the storage software stack is configured to use the additional logical volume metadata to access the second logical volume for I/O operations.

14. The system as recited in Claim 12, further comprising a second host, wherein the off-host virtualizer is further configured to:
generate second operating system metadata for a second virtual storage device, wherein the second virtual storage device is unmapped to physical storage;
make the second operating system metadata accessible to the second host;
aggregate a second range of physical storage within the first physical storage device into a second logical volume; and
provide second logical volume metadata for the second logical volume to the second host;
wherein the second host comprises a second storage software stack comprising a first and a second layer, wherein the first layer of the second software stack is configured to use the second operating system metadata to detect an existence of the second virtual storage device as a second addressable storage device, and wherein the second layer of the second software stack is configured to use the second logical volume metadata to access the second logical volume for I/O operations.

15. The system as recited in Claim 14, wherein the off-host virtualizer is further configured to prevent the second host from performing an I/O operation on the first logical volume.

16. The system as recited in Claim 14, wherein the first addressable storage device is accessible to the first host via a first storage network, and the second addressable storage device is accessible to the second host via a second storage network.

17. The system as recited in Claim 12, further comprising a second host and a second physical storage device, wherein the off-host virtualizer is further configured to:
generate second operating system metadata for a second virtual storage device, wherein the second virtual storage device is unmapped to physical storage;
make the second operating system metadata accessible to the second host;
aggregate a second range of physical storage within the second physical storage device into a second logical volume;
dynamically map a second range of virtual storage within the first virtual storage device to the second logical volume; and
provide second logical volume metadata for the second logical volume to the second host;
wherein the second host comprises a second storage software stack comprising a first and a second layer, wherein the first layer of the second software stack is configured to use the second operating system metadata to detect an existence of the second virtual storage device as a second addressable storage device, and wherein the second layer of the second software stack is configured to use the second logical volume metadata to access the second logical volume for I/O operations;
wherein the first addressable storage device is accessible to the first host via a first storage network, the second addressable storage device is accessible to the second host via a second storage network, the first physical storage device is accessible to the off-host virtualizer via a third storage network, and the second physical storage device is accessible to the off-host virtualizer via a fourth storage network.
18. The system as recited in Claim 17, wherein each storage network of the first, second, third and fourth storage networks includes an independently configurable fibre channel fabric.

19. The system as recited in Claim 1, wherein the storage software stack includes a second layer, further comprising a first and a second physical storage device, wherein the off-host virtualizer is further configured to: aggregate a first range of storage within the first physical storage device and a second range of storage within the second physical storage device into a single logical volume; and provide logical volume metadata for the single logical volume to the second layer of the storage software stack;

wherein the second layer of the storage software stack is configured to use the logical volume metadata to access the single logical volume for I/O operations.

20. A virtualizer operable to generate operating system metadata for a storage device, wherein the operating system metadata emulates a storage volume hosted under a particular operating system.

21. A method comprising:
generating first operating system metadata for a first virtual storage device;
making the first operating system metadata for the first virtual storage device accessible to a first host; and using the first operating system metadata at a first layer of a storage software stack at the first host to detect an existence of the first virtual storage device as a first addressable storage device.

22. The method as recited in Claim 21, further comprising:
customizing the operating system metadata in response to a requirement of a particular operating system in use at the first host.

23. The method as recited in Claim 21, wherein the first virtual storage device is unmapped to physical storage.

24. The method as recited in Claim 23, further comprising:
dynamically mapping a first and a second physical storage device to a respective address range within the first virtual storage device.

25. The method as recited in Claim 23, further comprising:
aggregating a first range of physical storage within a first physical storage device into a first logical volume;
dynamically mapping a first range of virtual storage within the first virtual storage device to the first logical volume;
providing first logical volume metadata for the first logical volume to a second layer of the storage software stack; and
using the first logical volume metadata to access the first logical volume for I/O operations from the second layer of the storage software stack.
26. The method as recited in Claim 25, further comprising:
aggregating a second range of physical storage within the first physical storage device into a second logical volume;
dynamically mapping a second range of virtual storage within the first virtual storage device to the second logical volume;
providing additional logical volume metadata for the second logical volume to the second layer of the storage software stack; and
using the additional logical volume metadata at the second layer of the storage software stack to access the second logical volume for I/O operations.

27. The method as recited in Claim 25, further comprising:
generating second operating system metadata for a second virtual storage device, wherein the second virtual storage device is unmapped to physical storage;
making the second operating system metadata accessible to a second host;
using the second operating system metadata at a first layer of a storage software stack at the second host to detect an existence of the second virtual storage device as a second addressable storage device.
aggregating a second range of physical storage within the first physical storage device into a second logical volume;
dynamically mapping a second range of virtual storage within the second virtual storage device to the second logical volume;
providing second logical volume metadata for the second logical volume to a second layer of the storage software stack at the second host; and
using the second logical volume metadata to access the second logical volume for I/O operations from the second layer of the storage software stack at the second host.

28. The method as recited in Claim 27, further comprising:
preventing the second host from performing an I/O operation on the first logical volume.

29. The method as recited in Claim 25, further comprising:
generating second operating system metadata for a second virtual storage device, wherein the second virtual storage device is unmapped to physical storage;
making the second operating system metadata accessible to a second host;
using the second operating system metadata at a first layer of a storage software stack at the second host to detect an existence of the second virtual storage device as a second addressable storage device.
aggregating a second range of physical storage within a second physical storage device into a second logical volume at an off-host virtualizer;
dynamically mapping a second range of virtual storage within the second virtual storage device to the second logical volume;
providing second logical volume metadata for the second logical volume to a second layer of the storage
software stack at the second host; and
using the second logical volume metadata to access the second logical volume for I/O operations from the
second layer of the storage software stack at the second host;

wherein the first range of physical storage is aggregated into the first logical volume by the off host virtualizer; and
wherein the first addressable storage device is accessible to the first host via a first storage network, the second
addressable storage device is accessible to the second host via a second storage network, the first physical
storage device is accessible to an off-host virtualizer via a third storage network, and the second physical
storage device is accessible to the off-host virtualizer via a fourth storage network.

30. The method as recited in Claim 29, wherein each storage network of the first, second, third and fourth
storage networks includes an independently configurable fibre channel fabric.

31. The method as recited in Claim 23, further comprising:
aggregating a first range of storage within a first physical storage device and a second range of storage
within a second physical storage device into a single logical volume;
dynamically mapping a range of virtual storage within the virtual storage device to the second logical
volume;
providing logical volume metadata for the single logical volume to a second layer of the storage software
stack; and
using the logical volume metadata to access the single logical volume for I/O operations from the second
layer of the storage software stack.

32. A computer accessible medium comprising program instructions, wherein the instructions are executable
to:
generate first operating system metadata for a first virtual storage device;
make the first operating system metadata for the first virtual storage device accessible to a first host; and
use the first operating system metadata at a first layer of a storage software stack at the host to detect an
existence of the first virtual storage device as a first addressable storage device.

33. The computer accessible medium as recited in Claim 32, wherein the instructions are further executable to:
customize the operating system metadata in response to a requirement of a particular operating system in
use at the first host.

34. The computer accessible medium as recited in Claim 32, wherein the first virtual storage device is
unmapped to physical storage.

35. The computer accessible medium as recited in Claim 34, wherein the instructions are further executable to:
map a first and a second physical storage device to a respective address range within the first virtual storage
device.
36. The computer accessible medium as recited in Claim 34, wherein the instructions are further executable to:
aggregate a first range of physical storage within a first physical storage device into a first logical volume;
dynamically map a first range of virtual storage within the first virtual storage device to the first logical
volume;
provide first logical volume metadata for the first logical volume to a second layer of the storage software
stack; and
use the first logical volume metadata to access the first logical volume for I/O operations from the second
layer of the storage software stack.
100

Software Storage Stack 140A

Host 110

- File System Layer 112
- Disk Driver Layer 114
- Interconnect-Dependent I/O Driver Layer 116

Interconnect 130

- Interconnect-Dependent Firmware Layer 122
- Physical Storage Layer 124

Physical Block Device 120

FIG. 1a
Host 110

File System Layer 112
Intermediate Driver Layer 113
Disk Driver Layer 114
Interconnect-Dependent I/O Driver Layer 116

Interconnect 130

Off-Host Virtualizer 180

Interconnect-Dependent Firmware Layer 122
Physical Storage Layer 124
Physical Block Device 120

FIG. 1b
Receive O.S. metadata requirements 410

Generate O.S. metadata for virtual storage device 420

Make O.S. metadata accessible to host 430

At first layer of storage stack, use O.S. metadata to detect existence of virtual storage device as addressable storage device 440

FIG. 4
Receive O.S. metadata requirements 510

Generate O.S. metadata for virtual storage device 520

Aggregate physical storage into logical volume 530

Map logical volume to address range within virtual storage device 540

Provide O.S. metadata and logical volume metadata to host 550

At first layer of storage stack, use O.S. metadata to detect existence of virtual storage device as addressable storage device 560

At second layer of storage stack, use logical volume metadata to perform I/O operations on logical volume 570

FIG. 5
FIG. 6

OS Metadata Header 215
(H blocks)
0
H-1

Unmapped
Blocks 235,
To Be
Mapped
Dynamically
(X Blocks)

Unmapped
Virtual LUN (VLUN)
230

OS Metadata Trailer 225
(T blocks)
H+X-1
H+X+T-1
Receive O.S. metadata requirements 1505

Generate O.S. metadata for unmapped virtual storage device 1510

Make O.S. metadata accessible to host 1515

At first layer of storage stack, use O.S. metadata to detect existence of virtual storage device as addressable storage device 1520

Dynamically map physical storage to virtual storage device 1525

FIG. 15
Receive O.S. metadata requirements 1605

Generate O.S. metadata for unmapped virtual storage device 1610

Make O.S. metadata accessible to host 1615

At first layer of storage stack, use O.S. metadata to detect existence of virtual storage device as addressable storage device 1620

Aggregate physical storage into logical volume 1625

Dynamically map logical volume to address range within virtual storage device 1630

Provide logical volume metadata to second layer of storage stack 1635

At second layer of storage stack, use logical volume metadata to perform I/O operations on logical volume 1640

FIG. 16
21/21

Processor 1910

Memory 1920

Software Storage Stack 140B

Local Storage Devices 1930

Software Storage Stack 140B

Host 110

FIG. 19

Network Interface 1940

To Interconnect

Computer Accessible Medium 2000

Virtualization Software 2010

FIG. 20
A. CLASSIFICATION OF SUBJECT MATTER
   IPC 7 G06F3/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
   Minimum documentation searched (classification system followed by classification symbols)
   IPC 7 G06F

   Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

   Electronic database consulted during the international search (name of database and, where practical, search terms used)
   EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>page 2 - page 3</td>
<td>9-19, 23-31, 34-36</td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of box C.

Patent family members are listed in annex.

* Special categories of cited documents
  *A* document defining the general state of the art which is not considered to be of particular relevance
  *E* earlier document but published on or after the international filing date
  *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
  *O* document referring to an oral disclosure, use, exhibition or other means
  *P* document published prior to the international filing date but later than the priority date claimed
  *S* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
  *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
  *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
  *S* document member of the same patent family

Date of the actual completion of the international search

10 May 2005

Date of mailing of the international search report

20/05/2005

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HT Rijswijk
Tel. (+31-70) 340-2040, Tx 31 651 epo nl, Fax (+31-70) 340-3016

Authorized officer

Dewyn, T
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>WO 02/065249 A (CONFLUENCE NETWORKS, INC; GAJJAR, KUMAR; COLLINS, JIM; PATEL, DIPAM; M) 22 August 2002 (2002-08-22) the whole document</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US 2002/103889 A1 (MARKSON THOMAS ET AL) 1 August 2002 (2002-08-01) the whole document</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US 6 493 811 B1 (BLADES JERRY A ET AL) 10 December 2002 (2002-12-10) the whole document</td>
<td></td>
</tr>
<tr>
<td>Patent document cited in search report</td>
<td>Publication date</td>
<td>Patent family member(s)</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 02067529 A2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1370945 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1370950 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1368742 A2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2004532442 T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 200452383 T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 02065249 A2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 02065309 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 02065290 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 02065298 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2002156987 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2002188711 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2002174306 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2003037127 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6880062 B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2005027754 A1</td>
</tr>
<tr>
<td>US 2002103889 A1</td>
<td>01-08-2002</td>
<td>US 6779016 B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 0198906 A2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AT 265707 T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 769928 B2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 6918200 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CA 2376333 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CN 1373871 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DE 60010277 D1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DE 60010277 T2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 1206738 A2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2003507817 T</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TW 548554 B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 0114987 A2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6714980 B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2003126265 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6597956 B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2003154279 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2002052941 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AU 7361701 A</td>
</tr>
</tbody>
</table>