(12) UK Patent Application (19) GB (11) 2519534 (13) A

(43) Date of A Publication 29.04.2015

(21) Application No: 1318712.5

(22) Date of Filing: 23.10.2013

(71) Applicant(s):
International Business Machines Corporation
New Orchard Road, Armonk 10504, New York, United States of America

(72) Inventor(s):
Martin Leo Schmatz
Animesh Kumar Trivedi
Bernard Metzler
Patrick Stuedi

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

(51) INT CL:
G06F 12/10 (2006.01) ***G06F 12/06*** (2006.01)
G06F 12/08 (2006.01) ***G06F 15/173*** (2006.01)

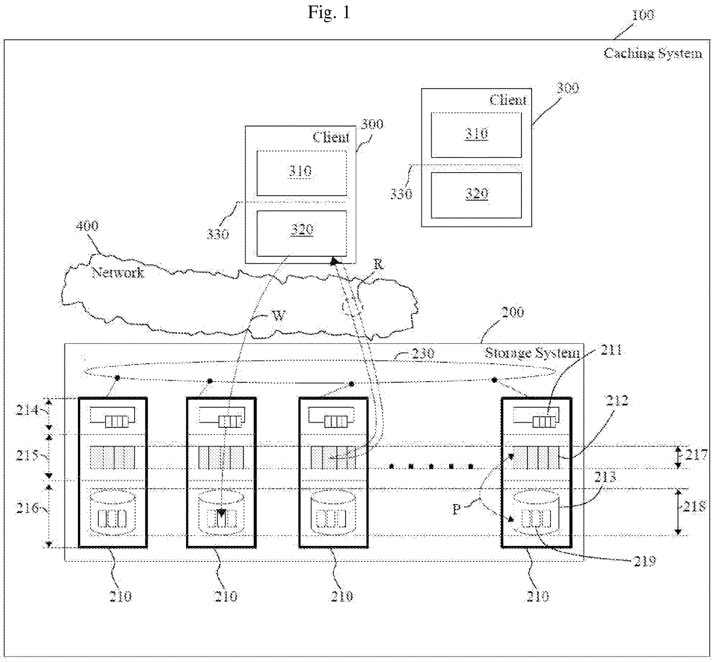
(56) Documents Cited:
EP 2150019 A1 **US 8255922 B1**
US 6598143 B1 **US 20130227201 A1**
US 20090287902 A1 **US 20070124407 A1**

(58) Field of Search:
INT CL **G06F**
Other: **WPI, EPODOC, XPI3E, INSPEC and TXTE**

(54) Title of the Invention: **Persistent caching system and method for operating a persistent caching system**
Abstract Title: **Persistent caching system utilising RDMA to request data from memory-mapped files**

(57) The application discloses a persistent caching system which includes a storage system having at least one caching server for storing data, and clients for accessing the data through a network. The caching server is configured to store the data in a number of virtual memory blocks, each of the virtual memory blocks referring to an associated memory-mapped file in a file system of the caching server. Further, the caching server is configured to export addresses of the virtual memory blocks to each of the clients. Each of the clients is configured to access at least some of the virtual memory blocks through Remote Direct Memory Access (RDMA) using at least some of the exported addresses. The caching server is further configured to page one or more virtual memory blocks being accessed by one or more clients through RDMA to and/or from the memory-mapped files associated with the accessed virtual memory blocks.

Fig. 1



GB 2519534 A

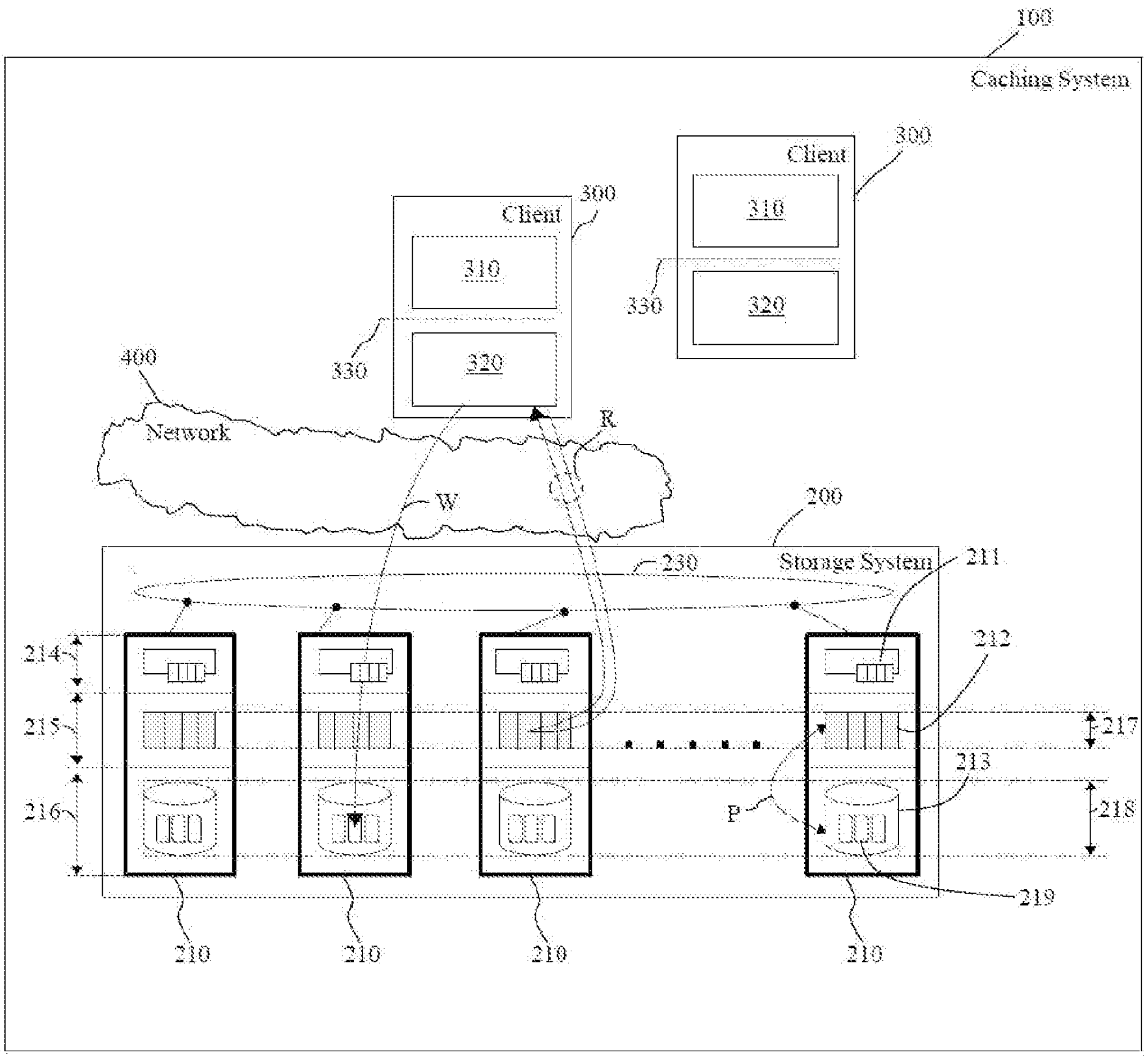

100
Caching System
Client
300
310
320
330
Client
300
310
320
330
400
Network
R
W
200
Storage System
230
211
212
213
214
215
216
217
218
219
P
210
210
210
210

Fig. 1

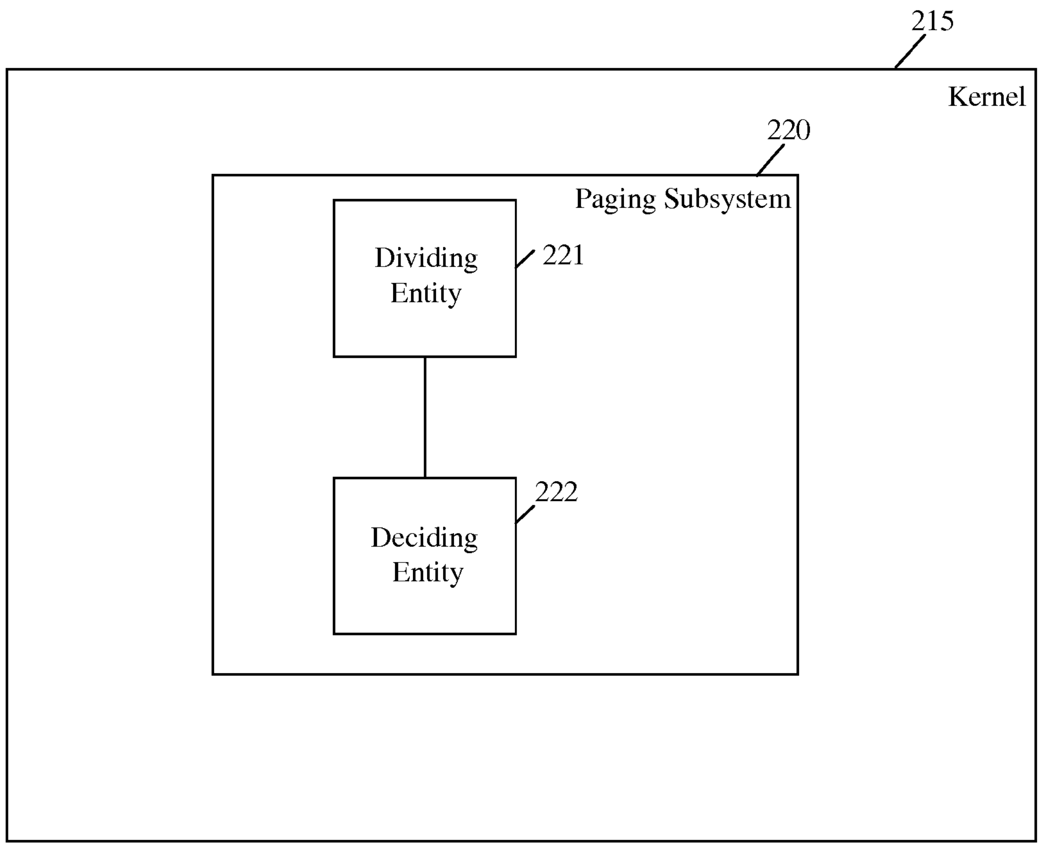

215
Kernel
220
Paging Subsystem
Dividing Entity
221
Deciding Entity
222

Fig. 2

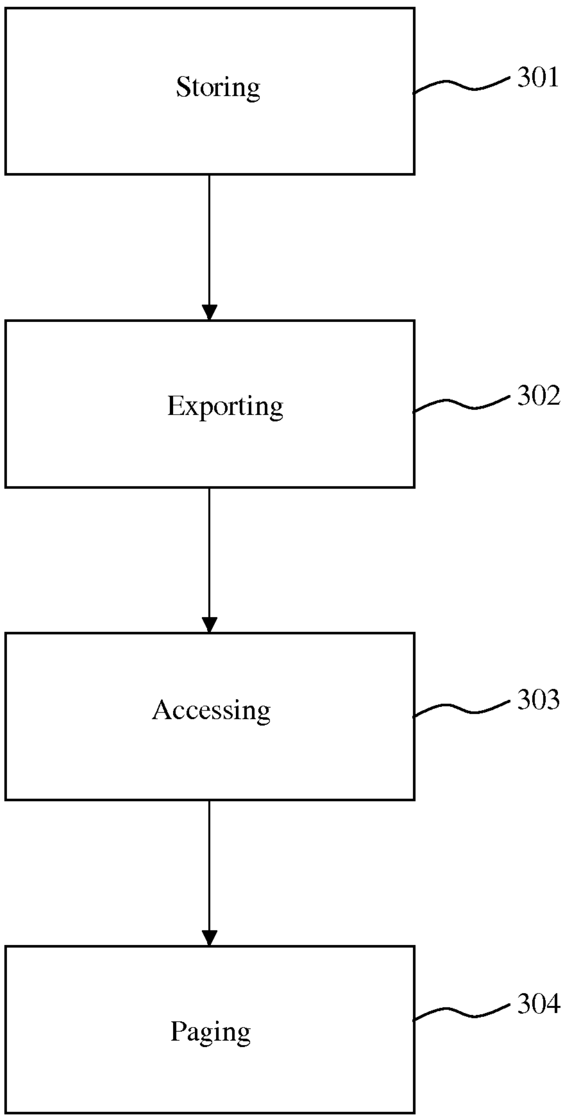


Fig. 3

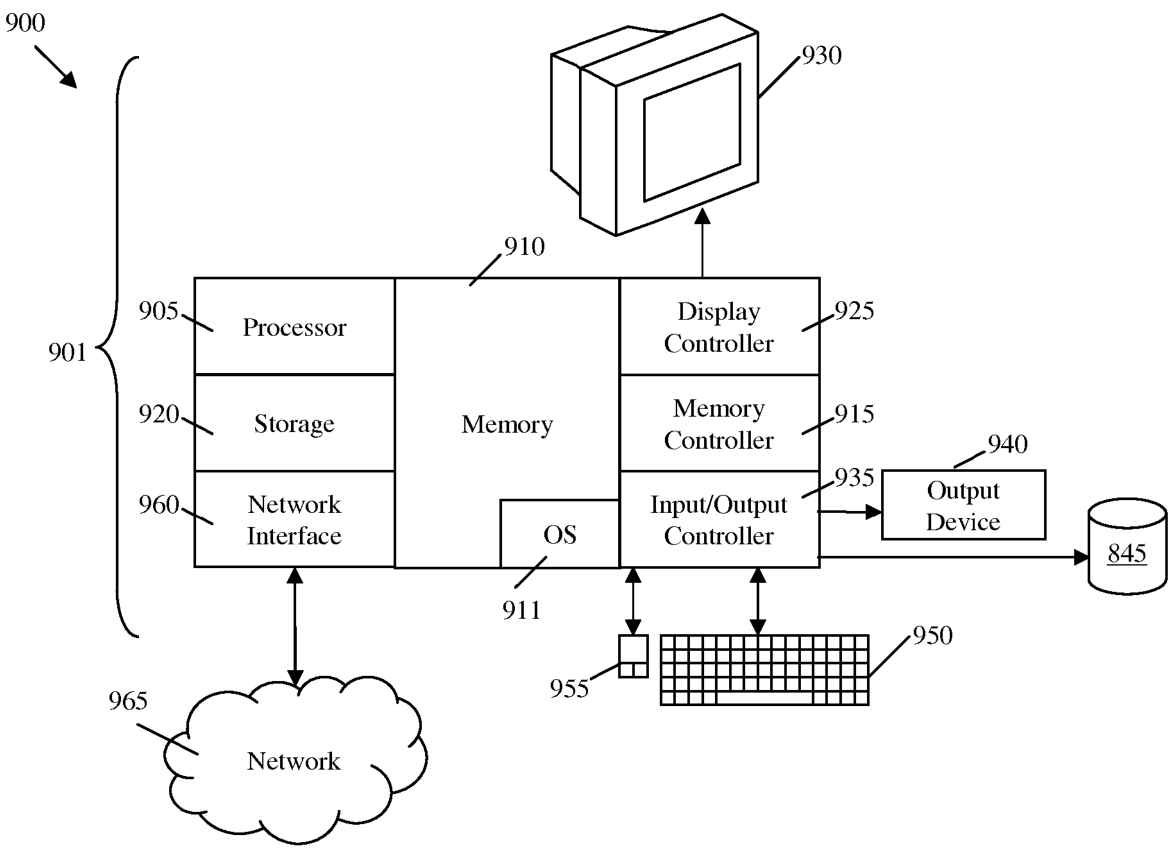

900
901
930
910
905
Processor
920
Storage
960
Network Interface
Memory
OS
911
Display Controller
925
Memory Controller
915
Input/Output Controller
935
Output Device
940
845
955
950
965
Network

Fig. 4

PERSISTENT CACHING SYSTEM AND METHOD FOR OPERATING A PERSISTENT CACHING SYSTEM

FIELD OF THE INVENTION

The invention relates to a persistent caching system and to a method for operating a persistent caching system using RDMA (Remote Direct Memory Access).

BACKGROUND

RDMA is a communication paradigm whereby application data is fetched directly out of a computer's local application memory and directly placed into the application memory of a remote computer. In bypassing the operating system and avoiding intermediate data copies in host memory, RDMA significantly reduces the CPU cost of large data transfers. Complete data copy avoidance (zero-copy) is achieved if the network interface controller (NIC) is able to move networked data directly between the application (buffer) memory and NIC buffer using a DMA engine.

Document US 2011/0078410 A1 describes a method of and a system for multiple party communications in a processing system including multiple processing subsystems. Each of the processing subsystems includes a central processing unit and one or more network adapters for connecting said each processing subsystem to the other processing subsystems. A multitude of nodes is established or created, and each of these nodes is associated with one of the processing subsystems. Here, pipelined communication using RDMA among three nodes may be involved, wherein the first node breaks up a large communication into multiple parts and sends these parts one after the other to the second node using RDMA, and the second node in turn absorbs and forwards each of these parts to a third node before all parts of the communication arrive from the first node.

The ever increasing amount of data stored and processed in data centers poses huge challenges not only to the data processing itself, but also in terms of power consumption and latency requirements. In particular, analytics and large-scale web applications running in data centers have stringent latency requirements.

To accommodate the latency requirements, applications may try to reduce data access latencies by storing much of the data in DRAM-based storage systems.

Disadvantageously, there is not enough DRAM to keep all the data in memory all the time. Further, availability is not guaranteed.

In reference [1], it is described to maintain a separate distributed cache close to the application, e.g. memcached, to cache the hot data. But the application has to handle consistency between the cache and persistent data. Memcached is described in reference [2].

Moreover, in-memory storages are known. In an in-memory storage, all the data is put inside a large distributed DRAM-based storage system with guaranteed availability. Disadvantageously, a huge amount of DRAM and special battery-backed buffers are required.

Therefore, it is an aspect of the present invention to provide an improved system for caching data.

BRIEF SUMMARY OF THE INVENTION

According to a first aspect, a persistent caching system is suggested. The persistent caching system includes a storage system having at least one caching server for storing data, and clients configured for accessing the data through a network. The caching server is configured to store the data in a number of virtual memory blocks, wherein each of the virtual memory blocks refers to an associated memory-mapped file in a file system of the caching server. Further, the caching server is configured to export addresses (virtual addresses) of the virtual memory blocks to each of the clients. Each of the clients is configured to access at least some of the virtual memory blocks through RDMA using at least some of the exported addresses. The caching server is further configured to page one or more virtual memory blocks being accessed by one or more clients through RDMA to and/or from the memory-mapped files associated with the accessed virtual memory blocks.

Advantageously, the present persistent caching system appears to the clients like a storage system. In particular, there is no need to maintain consistency between physical memory (cache) and persistence storage (disk). The present persistent caching system provides access to hot data with ultra-low latencies due to RDMA read operations from physical memory.

Moreover, the persistent caching system may hold data bigger than cluster DRAM size. Further, easy recovery of data from the mapped files is possible. Namely, each of the caching servers can be shut down and re-started without losing data. A caching server when started may create the virtual memory block from the memory-mapped files on a disk containing the persistent data.

According to some implementations, the present persistent caching system may use RDMA lazy memory pinning and memory mapped files to build distributed in-memory storage directly from physical memory, e.g. operating system page caches of cluster nodes. By means of the memory-mapped files, it is possible to access files through memory. On memory access, the operating system may bring disk blocks into page cache on-demand. Further, the operating system may swap pages of the page cache into reference-mapped files. Thus, the memory may be recovered easily from files. A memory-mapped file is a segment of virtual memory which is assigned a direct byte-for-byte correlation with some portion of a file or file-like resource.

According to some implementations, the present persistent caching system provides paging in network-based shared nothing memory architecture.

According to some implementations, paging may be understood as swapping. Swapping virtual memory blocks to and/or from persistent files means either writing data from physical memory to persistent files, or reading data from persistent files into physical memory.

In an embodiment, the storage system includes a plurality of caching servers configured to store the data. Each of the caching servers is configured to store at least a part of the data in a number of virtual memory blocks, wherein each of the virtual memory blocks refers to an associated memory-mapped file in a file system of the caching server. Each of the caching servers is further configured to export addresses of the virtual memory blocks to each of the clients. Each of the caching servers is further configured to page one or more virtual memory blocks being accessed by one or more clients through RDMA to and/or from the memory-mapped files associated with the accessed virtual memory blocks.

Advantageously, the storage system includes a huge amount of different caching servers which may be interconnected by a certain interconnect.

Because of including a plurality of caching servers, the persistent caching system is embodied as a persistent distributed caching system.

In a further embodiment, the caching server is configured to serve RDMA read operations from the clients directly from its operating system page cache.

In this embodiment, a direct service for RDMA read operations is provided at the level of the operating system of the caching server.
In a further embodiment, the caching server is configured to serve RDMA read operations from the clients directly from its operating system page cache using an in-kernel soft-RDMA stack or using hardware supported RDMA.

For example, each of the clients is configured to access the virtual memory blocks through soft-RDMA using the exported addresses.

In a further embodiment, each of the clients includes a client application and a storage library. The client application is linked to the storage library for accessing the storage system.

In particular, the client application uses the storage library to write data to and to read data from the plurality of caching servers, which may be called caching service. In detail, the storage library may provide two operations: write and read.

The write operation sends data to a caching server. The caching server may then take two actions: First, it stores (or copies) the data inside the virtual memory block. Second, it asks the kernel to page the virtual memory block out to the memory-mapped file. After the second step, the virtual memory block is persistently stored on disk.

The read operation may use RDMA to access the virtual address of the memory block. Namely, it uses an RDMA read to read parts of the virtual memory block. The RDMA read request may be received by a network card at the caching server. The network card may issue a DMA operation to copy the requested virtual memory block to the network card. From there, the virtual memory block may be transmitted back to the client.

During the processing of write requests at the caching server, the kernel, in particular the paging subsystem of the caching server, may be involved, namely pages of the virtual memory block being accessed may have to be paged in from disk, or paged out to disk.

During the processing of read requests at the caching server, the paging subsystem of the kernel of the caching server may be involved, namely pages of the virtual memory block may have to be paged in from disk.

In a further embodiment, the storage library is adapted to communicate with the at least one caching server of the storage system when writing data to or reading data from the storage system.

In a further embodiment, the client applications and the caching servers are adapted to run on a disjoint set of compute nodes in the network or on a common set of compute nodes in the network.

Alternatively, the client applications and the caching servers may run on an overlapping set of compute nodes.

In a further embodiment, the caching server is configured to create the virtual memory blocks using the memory-mapped files.

The terms virtual memory blocks and memory-mapped files may be used interchangeably, e.g. each virtual memory block is referring to a memory-mapped file.

In a further embodiment, the caching server is configured to store the data of one of the virtual memory blocks in a physical memory or on a disk in the associated memory-mapped file at any given time.

Thus, the caching server is adapted to page or swap the data of one of the virtual memory blocks at its own decision at any given time.

In a further embodiment, a kernel of the caching server includes a paging subsystem which is configured to decide, at any given time, on storing the data of one of the virtual memory blocks in the physical memory or on the disk in the memory-mapped file.
Thus, for each virtual memory block, the decision about when the memory block should be present in physical memory, and when the memory block should be held on disk in the file, is made by the kernel of the caching server, namely by the paging subsystem of the kernel.

In a further embodiment, the paging subsystem is configured to keep more frequently accessed virtual memory blocks present in the physical memory and less frequently accessed virtual memory blocks on the disk in the memory-mapped file.
Thus, the more frequently accessed virtual memory blocks are provided faster than less frequently accessed memory blocks. Thus, the overall latency is decreased.

In a further embodiment, the paging subsystem includes a dividing entity and a deciding entity, wherein the dividing entity is configured to subdivide a virtual memory block into a plurality of pages, wherein the deciding entity is configured to decide on keeping a first part of the pages in the physical memory and a second part of the pages on the disk.
A page may be 4K for example. The deciding entity may decide to keep parts of the virtual memory block, e.g. some pages, in physical memory and some other pages on disk.

In a further embodiment, the storage library is configured to provide a write operation through RDMA for writing data into the storage subsystem and a read operation through RDMA for reading data from the storage system.

In a further embodiment, after receiving data from the storage library of one of the clients, the caching server is configured to store the received data in the number of virtual memory blocks and to ask the kernel to page the number of virtual memory blocks out to the associated memory-mapped file.

In a further embodiment, the paging subsystem is configured to page at least one page of the virtual memory block being accessed by the write operation in from the disk and to page it later out to the disk.

In a further embodiment, after receiving an RDMA read request from a requesting client at a network card of the caching server, the network card is configured to issue a DMA operation to copy the virtual memory block requested by the RDMA read request to the network card and to transmit the copied virtual memory block to the requesting client.

In a further embodiment, the paging subsystem is configured to page at least one page of the virtual memory block being accessed by the read operation in from the disk.

In a further embodiment, access rights are allocated to at least one of the memory-mapped files. The allocated access rights determine which clients are allowed to access the at least one of the memory-mapped file.

Any embodiment of the first aspect may be combined with any embodiment of the first aspect to obtain another embodiment of the second aspect.

According to a second aspect, the invention can be embodied as a method for operating a persistent caching system including a storage system having at least one caching server for storing data, and clients configured for accessing the data through a network. In a first step, the data is stored in a number of virtual memory blocks by the caching server, wherein each of the virtual memory blocks refers to an associated memory-mapped file in a file system of the caching server. In a second step, addresses of the virtual memory blocks are exported from the caching server to each of the clients. In a third step, at least some of the virtual memory blocks are accessed by at least one of the clients through RDMA using at least some of the exported addresses. In a fourth step, one or more virtual memory blocks being accessed by one or more clients through RDMA are paged to and/or from the memory-mapped files associated with the accessed virtual memory blocks.

According to a third aspect, a computer program is suggested which comprises a program code for executing the method of the above second aspect for operating a persistent caching system when run on at least one computer.

In the following, exemplary embodiments of the present invention are described with reference to the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows a schematic block diagram of an embodiment of a persistent caching system;

Fig. 2 shows a schematic block diagram of an extract of a kernel of a caching server of the persistent caching system of Fig. 1;

Fig. 3 shows an embodiment of a sequence of method steps for operating a persistent caching system; and

Fig. 4 shows a schematic block diagram of an embodiment of a system adapted for data transmissions using RDMA.

Similar or functionally similar elements in the figures have been allocated the same reference signs if not otherwise indicated.

DETAILED DESCRIPTION OF THE EMBODIMENTS

In Fig. 1, a schematic block diagram of an embodiment of a persistent caching system 100 is depicted. The persistent caching system 100 of Fig. 1 includes a number of clients 300 and a storage system 200. Without loss of generality, Fig. 1 shows two clients 300.

The storage system 200 comprises a plurality of caching servers 210 which may be coupled by an interconnect 230. The clients 300 and the storage system 200 may be connected by or may be part of a network 400, e.g. the internet or a local area network (LAN). Each of the caching servers 210 includes a user space 214, a kernel 215 and an I/O space 216. Because the caching servers 210 are interconnected to one storage system 200, the clients 300 may see a distributed physical memory 217 including the physical memories 212 of the caching servers 210 and an effective storage space 218 including the disks 213 of the caching servers 210. The respective disk 213 includes files 219 holding persistent data. Without loss of generality, Fig. 1 shows a plurality of caching servers 210, e.g. four caching servers 210.

As indicated above, the clients 300 are adapted to access (at least some of) the data stored in the storage system 200 through the network 400. Each of the clients 300 may for instance include a client application 310 and a storage library 320. The client application 310 is linked to the storage library 320 for accessing the storage system 200. In particular, the storage library 320 is adapted to communicate with the caching servers 210 of the storage system 200 on writing data to or reading data from the storage system 200.

As depicted in Fig. 1, the client applications 310 and the caching servers 210 are adapted to run on a disjoint set of computing nodes in the network 400. Alternatively, the client applications 310 and the caching servers 210 may be adapted to run on a common set of computer nodes (not shown).

Each caching server 210 is adapted to store the data in a number of virtual memory blocks, wherein each of the virtual memory blocks refers to an associated memory-mapped file 211 in a file system of the caching server 210. Further, the respective caching server 210 is adapted to export addresses (virtual addresses) of the virtual memory blocks to each of the clients 300. As a result, each of the clients 300 is adapted to access (at least some of) the virtual memory blocks through RDMA R, W using (at least some of) the exported addresses. Using RDMA includes RDMA read operations R and RDMA write operations W.

Moreover, the respective caching server 210 is configured to page P one or more virtual memory blocks being accessed by one or more clients 300 through RDMA R, W to and/or from the memory-mapped files 211 associated with the accessed virtual memory blocks.

The persistent caching system 100 appears to the clients 300 like a storage system. In particular, there is no need to maintain consistency between physical memory 212 (cache) and persistence storage 213 (disk). The persistent caching system 100 may provide access to hot data with ultra-low latencies due to RDMA read operations from physical memory 212.

Furthermore, the persistent caching system 100 may hold data bigger than cluster DRAM size. Further, easy recovery of data from the memory-mapped files 211 is possible. Namely, each of the caching servers 210 may be shut down and re-started without losing data. A caching server 210 when started may create the virtual memory block from the memory-mapped files 211 on a disk 213 containing the persistent data.

Note that the data (e.g., stored as files), may have access rights, owners, etc. Thus, clients may be allowed to access only certain files (and blocks thereof). To that aim, access rights can be allocated to some of the memory-mapped files. The allocated access rights determine which clients are allowed to access which the memory-mapped files. For example, a first client may be able to access data *A* though a subset *a* of the export addresses, while not being able to access data *B*, whereas a second client may access data *B* though another subset *b* of the export addresses (while not necessarily able to access data *A*, etc.). For this reason and most generally, each client is adapted to access *at least some of* the virtual memory blocks through RDMA using *at least some of* the exported addresses. This distinction is not reiterated in the following. Without prejudice, it is hereafter assumed that clients can access the same data using the same exported addresses (made available to them all), for simplicity.

Next, the respective caching server 210 may be adapted to serve RDMA read operations R from the clients 300 directly from its operating system page cache 212. For example, the caching server 210 may be configured to serve said RDMA read operations R from the clients 300 directly from its operating system page cache 212 using an in-kernel soft-RDMA stack or using hardware-supported RDMA. In particular, the respective caching server 210 is configured to create the virtual memory blocks using said memory-mapped files 211. At any given time, the respective caching server 210 may be adapted to store the data of one of the virtual memory blocks in the physical memory 212 or on the disk 213 in the associated memory-mapped file.

The storage library 320 of the respective client 300 may be configured to provide and to transmit a write operation W for writing data into the storage system 210 and a read operation R through RDMA for reading data from the storage system 210.

In the case of a write operation W, after receiving data from the storage library 320 of one of the clients 300, the caching server 210 stores the received data in the number of virtual memory blocks and asks the kernel 215 to page P the number of virtual memory blocks out to the associated memory-mapped file 211. In this regard, the paging subsystem 220 may page P at least one page of the virtual memory block being accessed by the write operation W in from the disk 213 and to page it later out to the disk 213.

In case of a read operation R, after receiving an RDMA read request R from a requesting client 300 at a network card of the caching server 210, the network card issues a DMA operation to copy the virtual memory block requested by the RDMA read request R to said network card and to transmit the copied virtual memory block to the requesting client 300. In this case, the paging subsystem 220 may page P at least one page of the virtual memory block being accessed by said read operation R in from the disk 213.

Fig. 2 shows a schematic block diagram of an extract of a kernel 215 of one caching server 210 of the persistent caching system 100 of Fig. 1. The kernel 215 of the caching server 210 includes a paging subsystem 220. The paging subsystem 220 is adapted to decide, at any given time, on storing the data of the respective virtual memory block in the physical memory 212 or on the disk 213 in the memory-mapped file. Furthermore, the paging subsystem 220 is configured to keep more frequently accessed virtual memory blocks present in the physical memory 212, but less frequently accessed memory blocks on the disk 213 in the memory-mapped file.

As shown in Fig. 2, the paging subsystem 212 includes a dividing entity 221 and a deciding entity 222. The dividing entity 221 is adapted to subdivide a virtual memory block into a plurality of pages. Further, the deciding entity 222 is configured to decide on keeping a first part of the pages in the physical memory 212 and a second part of the pages on the disk 213. Also, the ratio of the first and second parts may be decided by said deciding entity 222.

Fig. 3 shows an embodiment of a sequence of method steps for operating a persistent caching system.

In Fig. 3, an embodiment of a sequence of method steps for operating a persistent caching system 100 is shown. The persistent caching system 100 may be embodied as shown in Fig. 1. The caching system 100 includes a storage system 200 having a number of caching servers 210 for storing data, and clients 300 for accessing the data through a network 400.

The method of Fig. 3 includes the following steps 301 to 304:

In step 301, the data is stored in a number of virtual memory blocks by at least one of the caching servers. Therein, each of the virtual memory blocks refers to an associated memory-mapped file in a file system of the caching server.

In step 302, addresses, in particular virtual addresses, of the virtual memory blocks are exported from the at least one caching server to each of the clients.

In step 303, the virtual memory blocks are accessed by at least one of the clients through RDMA using the exported addresses.

In step 304, one or more virtual memory blocks accessed by one or more clients through RDMA are paged to and/or from the memory-mapped files associated with the accessed virtual memory blocks.

Computerized devices can be suitably designed for implementing embodiments of the present invention as described herein. In that respect, it can be appreciated that the methods described herein are largely non-interactive and automated. In exemplary embodiments, the methods described herein can be implemented either in an interactive, partly-interactive or non-interactive system. The methods described herein can be implemented in software (e.g., firmware), hardware, or a combination thereof. In exemplary embodiments, the methods described herein are implemented in software, as an executable program, the latter executed by suitable digital processing devices. In further exemplary embodiments, at least one step or all steps of above method of Fig. 3 may be implemented in software, as an executable program, the latter executed by suitable digital processing devices. More generally, embodiments of the present invention can be implemented wherein general-purpose digital computers, such as personal computers, workstations, etc., are used.

For instance, the system 900 depicted in Fig. 9 schematically represents a computerized unit 901, e.g., a general-purpose computer. In exemplary embodiments, in terms of hardware architecture, as shown in Fig. 9, the unit 901 includes a processor 905, memory 910 coupled to a memory controller 915, and one or more input and/or output (I/O) devices 940, 945, 950, 955 (or peripherals) that are communicatively coupled via a local input/output controller 935. The input/output controller 935 can be, but is not limited to, one or more buses or other wired or wireless connections, as is known in the art. The input/output controller 935 may have additional elements, which are omitted for simplicity, such as controllers, buffers (caches), drivers, repeaters, and receivers, to enable communications. Further, the local interface may include address, control, and/or data connections to enable appropriate communications among the aforementioned components.

The processor 905 is a hardware device for executing software, particularly that stored in memory 910. The processor 905 can be any custom made or commercially available processor, a central processing unit (CPU), an auxiliary processor among several processors associated with the computer 901, a semiconductor based microprocessor (in the form of a microchip or chip set), or generally any device for executing software instructions.

The memory 910 can include any one or combination of volatile memory elements (e.g., random access memory) and nonvolatile memory elements. Moreover, the memory 910 may incorporate electronic, magnetic, optical, and/or other types of storage media. Note that the memory 910 can have a distributed architecture, where various components are situated remote from one another, but can be accessed by the processor 905.

The software in memory 910 may include one or more separate programs, each of which comprises an ordered listing of executable instructions for implementing logical functions. In the example of Fig. 9, the software in the memory 910 includes methods described herein in accordance with exemplary embodiments and a suitable operating system (OS) 911. The OS 911 essentially controls the execution of other computer programs, such as the method as described herein (e.g., Fig. 3), and provides scheduling, input-output control, file and data management, memory management, and communication control and related services.

The methods described herein may be in the form of a source program, executable program (object code), script, or any other entity comprising a set of instructions to be performed. When in a source program form, then the program needs to be translated via a compiler, assembler, interpreter, or the like, as known per se, which may or may not be included within the memory 910, so as to operate properly in connection with the OS 911. Furthermore, the methods can be written as an object oriented programming language, which has classes of data and methods, or a procedure programming language, which has routines, subroutines, and/or functions.

Possibly, a conventional keyboard 950 and mouse 955 can be coupled to the input/output controller 935. Other I/O devices 940 – 955 may include sensors (especially in the case of network elements), i.e., hardware devices that produce a measurable response to a change in a physical condition like temperature or pressure (physical data to be monitored). Typically, the analog signal produced by the sensors is digitized by an analog-to-digital converter and sent to controllers 935 for further processing. Sensor nodes are ideally small, consume low energy, are autonomous and operate unattended.

In addition, the I/O devices 940 – 955 may further include devices that communicate both inputs and outputs. The system 900 can further include a display controller 925 coupled to a display 930. In exemplary embodiments, the system 900 can further include a network interface or transceiver 960 for coupling to a network 965.

The network 965 transmits and receives data between the unit 901 and external systems. The network 965 is possibly implemented in a wireless fashion, e.g., using wireless protocols and technologies, such as WiFi, WiMax, etc. The network 965 may be a fixed wireless network, a wireless local area network (LAN), a wireless wide area network (WAN) a personal area network (PAN), a virtual private network (VPN), intranet or other suitable network system and includes equipment for receiving and transmitting signals.

The network 965 can also be an IP-based network for communication between the unit 901 and any external server, client and the like via a broadband connection. In exemplary embodiments, network 965 can be a managed IP network administered by a service provider. Besides, the network 965 can be a packet-switched network such as a LAN, WAN, Internet network, etc.

If the unit 901 is a PC, workstation, intelligent device or the like, the software in the memory 910 may further include a basic input output system (BIOS). The BIOS is stored in ROM so that the BIOS can be executed when the computer 901 is activated.

When the unit 901 is in operation, the processor 905 is configured to execute software stored within the memory 910, to communicate data to and from the memory 910, and to generally control operations of the computer 901 pursuant to the software. The methods described herein and the OS 911, in whole or in part are read by the processor 905, typically buffered within the processor 905, and then executed. When the methods described herein (e.g. with reference to Fig. 3) are implemented in software, the methods can be stored on any computer readable medium, such as storage 920, for use by or in connection with any computer related system or method.

As will be appreciated by one skilled in the art, aspects of the present invention may be embodied as a system, method or computer program product. Accordingly, aspects of the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects. Furthermore, aspects of the present invention may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon. Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disk read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

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

Computer program code for carrying out operations for aspects of the present invention may be written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the unit 901, partly thereon, partly on a unit 901 and another unit 901, similar or not.

Aspects of the present invention are described above with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams can be implemented by one or more computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

The flowchart and block diagrams in the figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the blocks may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved and algorithm optimization. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

More generally, while the present invention has been described with reference to certain embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the present invention. In addition, many modifications may be made to adapt a particular situation to the teachings of the present invention without departing from its scope. Therefore, it is intended that the present invention not be limited to the particular embodiments disclosed, but that the present invention will include all embodiments falling within the scope of the appended claims.

REFERENCE SIGN LIST

100	caching system
200	storage system
210	caching server
211	memory-mapped file
212	physical memory
213	disk
214	user space
215	kernel
216	I/O space
217	distributed physical memory
218	effected storage space of the storage system
219	file holding persistent data
220	paging subsystem
221	dividing entity
222	deciding entity
230	interconnect
300	client
301	method step
302	method step
303	method step
304	method step
310	client application
320	storage library
330	storage API
P	paging
R	RDMA read operation
W	RDMA write operation

REFERENCES

[1] Patrick Stuedi, Animesh Trivedi, Bernard Metzler; IBM Research, Zurich: Wimpy Nodes with 10GbE: Leveraging One-Sided Operations in Soft-RDMA to Boost Memcache

[2] Memcached - a distributed memory object caching system; http://memcached.org.

CLAIMS

1. A persistent caching system (100), comprising:

a storage system (200) having at least one caching server (210) for storing data, and

clients (300) configured for accessing at least some of the data through a network (400), wherein the caching server (210) is configured to:

store the data in a number of virtual memory blocks, each of the virtual memory blocks referring to an associated memory-mapped file (211) in a file system of the caching server (210); and

export addresses of the virtual memory blocks to each of the clients (300),

wherein each of the clients (300) is configured to access at least some of the virtual memory blocks through RDMA (R, W) using at least some of the exported addresses, and

wherein the caching server (210) is further configured to page (P) one or more virtual memory blocks being accessed by one or more clients (300) through RDMA (R, W) to and/or from the memory-mapped files (211) associated with the accessed virtual memory blocks.

2. The persistent caching system of claim 1,
wherein the storage system (200) includes a plurality of caching servers (210) configured to store the data, wherein each of the caching servers (210) is configured to store at least a part of the data in a number of virtual memory blocks, each of the virtual memory blocks referring to an associated memory-mapped file (211) in a file system of the caching server (210), and to export addresses of the virtual memory blocks to each of the clients (300), wherein each of the caching servers (210) is further configured to page (P) one or more virtual memory blocks being accessed by one or more clients through RDMA (R, W) to and/or from the memory-mapped files (211) associated to the accessed virtual memory blocks.

3. The persistent caching system of claim 1 or 2,
wherein the caching server (210) is configured to serve RDMA read operations (R) from the clients (300) directly from its operating system page cache (212).

4. The persistent caching system of claim 3,
wherein the caching server (210) is configured to serve RDMA read operations (R) from the clients (300) directly from its operating system page cache (212) using an in-kernel soft-RDMA stack or using hardware supported RDMA.

5. The persistent caching system of one of claims 1 to 4,
wherein each of the clients (300) includes a client application (310) and a storage library (320), wherein the client application (310) is linked to the storage library (320) for accessing the storage system (200).

6. The persistent caching system of one of claims 1 to 5,
wherein the caching server (210) is configured to create the virtual memory blocks using the memory-mapped files (211).

7. The persistent caching system of one of claims 1 to 6,
wherein the caching server (210) is configured to store the data of one of the virtual memory blocks in a physical memory (212) or on a disk (213) in the associated memory-mapped file (211) at any given time.

8. The persistent caching system of claim 7,
wherein a kernel (215) of the caching server (210) includes a paging subsystem (220) which is configured to decide, at any given time, on storing the data of one of the virtual memory blocks in the physical memory (212) or on the disk (213) in the memory-mapped file (211).

9. The persistent caching system of claim 8,
wherein the paging subsystem (220) is configured to keep more frequently accessed virtual memory blocks present in the physical memory (212) and less frequently accessed virtual memory blocks on the disk (213) in the memory-mapped file (211).

10. The persistent caching system of claim 8 or 9,
wherein the paging subsystem (220) includes a dividing entity (221) and a deciding entity (222), wherein the dividing entity (221) is configured to subdivide a virtual memory block into a plurality of pages, wherein the deciding entity (222) is configured to decide on keeping a first part of the pages in the physical memory (212) and a second part of the pages on the disk (213).

11. The persistent caching system of one of claims 5 to 10,

wherein the storage library (320) is configured to provide a write operation (W) for writing data into the storage system (200) and a read operation (R) through RDMA for reading data from the storage system (200).

12. The persistent caching system of claim 11,
wherein, after receiving data from the storage library (320) of one of the clients (300), the caching server (210) is configured to store the received data in the number of virtual memory blocks and to ask the kernel (215) to page (P) the number of virtual memory blocks out to the associated memory-mapped file (211).

13. The persistent caching system of claim 12,
wherein the paging subsystem (220) is configured to page (P) at least one page of the virtual memory block being accessed by the write operation (W) in from the disk (213) and to page it later out to the disk (213).

14. The persistent caching system of one of claims 11 to 13,
wherein, after receiving an RDMA read request (R) from a requesting client (300) at a network card of the caching server (210), the network card is configured to issue a DMA operation to copy the virtual memory block requested by the RDMA read request (R) to the network card and to transmit the copied virtual memory block to the requesting client (300).

15. A method for operating a persistent caching system including a storage system having at least one caching server for storing data, and clients configured for accessing at least some of the data through a network, the method comprising:

storing (301) the data in a number of virtual memory blocks by the caching server, each of the virtual memory blocks referring to an associated memory-mapped file in a file system of the caching server,

exporting (302) addresses of the virtual memory blocks from the caching server to each of the clients,

accessing (303) at least some of the virtual memory blocks by at least one of the clients through RDMA using at least some of the exported addresses, and

paging (304) one or more virtual memory blocks being accessed by one or more clients through RDMA to and/or from the memory-mapped files associated with the accessed virtual memory blocks.

Intellectual
Property
Office

Application No:	GB1318712.5	**Examiner:**	Dr Mark Edwards
Claims searched:	1-15	**Date of search:**	28 March 2014

Patents Act 1977: Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance
X	X: 1-15	US 2013/0227201 A1 (TALAGALA) See especially figures 2-4 and columns 72, 130-133, 151-158, 165 & 185-188
X	X: 1-15	US 2009/0287902 A1 (FULLERTON) See especially Figures 2-6 and columns 13-24, 29-40, 44-51 & 67
Y	Y1: 1-4, 6-10 & 15	US 2007/0124407 A1 (WEBER) See especially columns 12, 21, 45-53 & 60-63
Y	Y1: 1-4, 6-10 & 15	EP 2150019 A1 (LUCENT) Abstract, figures 2-3 and paragraphs 3-8 & 13-28
Y	Y2: 1-4, 6-10 & 15	US 6598143 B1 (BAKER) See especially Figure 4 and columns 50-67
Y	Y2: 1-4, 6-10 & 15	US 8255922 B1 (FRESKO) Abstract, figures 4-7, column 1 (lines 6-41), 2 (line 28)-3 (line 11)

Categories:

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC^{X} :

Worldwide search of patent documents classified in the following areas of the IPC

G06F

The following online and other databases have been used in the preparation of this search report

WPI, EPODOC, XPI3E, INSPEC and TXTE

International Classification:

Subclass	Subgroup	Valid From
G06F	0012/10	01/01/2006
G06F	0012/06	01/01/2006
G06F	0012/08	01/01/2006
G06F	0015/173	01/01/2006