



(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/EP2014/078498

(22) International Filing Date:
18 December 2014 (18.12.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14305046.6 13 January 2014 (13.01.2014) EP

(71) Applicant: **THOMSON LICENSING** [FR/FR]; 1-5 rue
Jeanne d'Arc, F-92130 Issy-les-Moulineaux (FR).

(72) Inventor: **ARRIGHETTI, Walter**; Deutsche Thomson
OHG, Karl-Wiechert-Allee 74, 30625 Hannover (DE).

(74) Agent: **SCHMIDT-UHLIG, Thomas**; Deutsche Thomson
OHG, European Patent Operations, Karl-Wiechert-Allee
74, 30625 Hannover (DE).

(81) Designated States (unless otherwise indicated, for every
kind of national protection available): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY,
BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM,
DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,
HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR,
KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG,
MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM,
PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC,
SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every
kind of regional protection available): ARIPO (BW, GH,
GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ,
TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU,
TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE,
DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU,
LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,
SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ,
GW, KM, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR HANDLING DATA IN A STORAGE SYSTEM

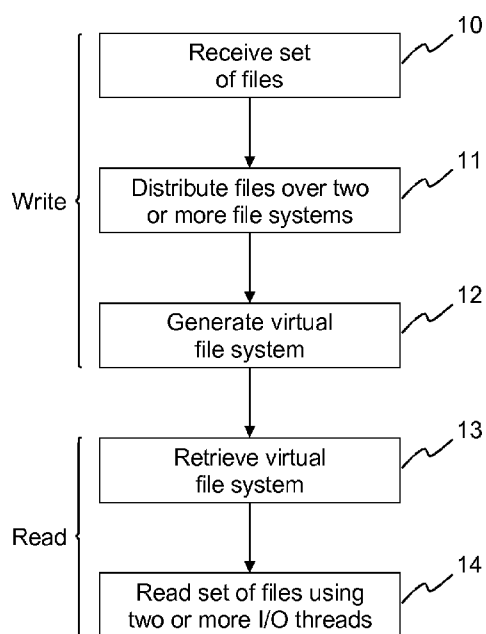


Fig. 1

(57) Abstract: A method for handling data in a storage system (25) and an apparatus (20) configured to perform the method are described. In particular, in order to store data in the storage system (25), the data being provided as a set of self-contained files, a file distributor (22) distributes (11) the files over two or more file systems based on one or more properties of the files. Each file system has an independent data path. A file system generator (23) generates (12) a virtual file system comprising pointers to the distributed files within the two or more file systems. For reading data from the storage system (25), a file system retrieving unit (26) retrieves (13) the virtual file system. A data interface (24) then reads (14) a set of files from the two or more file systems based on the virtual file system using two or more I/O threads.

WO 2015/104159 A1



Published:

— *with international search report (Art. 21(3))*

METHOD AND APPARATUS FOR HANDLING DATA IN A STORAGE SYSTEM**FIELD OF THE INVENTION**

5 The invention relates to a method and an apparatus for handling data in a storage system, and more specifically to a method and an apparatus for handling data that are provided as a set of self-contained files distributed over two or more file systems.

10 **BACKGROUND OF THE INVENTION**

In high-profile post-production for digital intermediates and digital cinema, high-resolution uncompressed video footage is handled. High resolution here means frames with at least 10
15 bits per color channel per pixel, thus a minimum of 32 bits per pixel (including padding bits), and a resolution which is at least 'HD' (1920 pixels per row), but is usually '2K' (2048 pixels per row), '4K' (4096 pixels per row), or even more.

20 In addition, a throughput equivalent to at least 24 frames per second is needed for real-time record and playback capabilities, which is mandatory in certain creative passages as well as in theatrical shows. This means that any storage system which is real-time capable must be carefully engineered
25 to guarantee those minimum requirements. For example, typical monoscopic theatrical 2K playback requires 305MB/s minimum throughput all over. Stereoscopic frames, with 12 bits rather than 10 bits per color channel, results in a minimum throughput of 765MB/s. Monoscopic theatrical 4K playback goes as up as
30 1.22GB/s, and so on.

Uncompressed video footage, but also video that is compressed using intra-frame compression schemes only, is traditionally stored as frame sequences, particularly during the post-

production and finishing phases. This means that each subsequent frame of a time-continuous video clip is stored as a single file; thus the term "frame-per-file" sequence. A finished 'movie' may originate from a final cut of several such clips. Maximizing the I/O performance when reading and writing video clips based on frame-per-file sequences is crucial for playing back and recording high-resolution, uncompressed video. The I/O performance is currently limited by hardware I/O throughput of the underlying storage device, e.g. the theoretical maximum of 30MB/s for USB 2.0 Direct-Attached Storage, as well as by file-allocation strategies of the file system and subsequent internal file fragmentation.

SUMMARY OF THE INVENTION

It is an object of the present invention to propose an improved solution for handling data in a storage system, the data being provided as a set of files.

According to one aspect of the invention, a method for storing data in a storage system, wherein the data are provided as a set of self-contained files, comprises the steps of:

- distributing the files over two or more file systems based on one or more properties of the files, each file system having an independent data path; and
- generating a virtual file system comprising pointers to the distributed files within the two or more file systems.

Accordingly, an apparatus configured to store data in a storage system, wherein the data are provided as a set of self-contained files, comprises:

- a file distributor configured to distribute the files over two or more file systems based on one or more properties of the files, each file system having an independent data path; and

- a file system generator configured to generate a virtual file system comprising pointers to the distributed files within the two or more file systems.

5 Also, a computer readable storage medium has stored therein instructions enabling storing data in a storage system, wherein the data are provided as a set of self-contained files, which when executed by a computer, cause the computer to:

- distribute the files over two or more file systems based on
10 one or more properties of the files, each file system having an independent data path; and

- generate a virtual file system comprising pointers to the distributed files within the two or more file systems.

15 According to another aspect of the invention, a method for reading data from a storage system, wherein the data are provided as a set of self-contained files distributed over two or more file systems, each file system having a data path, comprises the steps of:

20 - retrieving a virtual file system comprising pointers to the distributed files within the two or more file systems; and
- reading a set of files from the two or more file systems based on the virtual file system using two or more I/O threads.

25 Accordingly, an apparatus configured to read data from a storage system, wherein the data are provided as a set of self-contained files distributed over two or more file systems, each file system having a data path, comprises:

- a file system retrieving unit configured to retrieve a
30 virtual file system comprising pointers to the distributed files within the two or more file systems; and
- a data interface (configured to read a set of files from the two or more file systems based on the virtual file system using two or more I/O threads.

Similarly, a computer readable storage medium has stored therein instructions enabling reading data from a storage system, wherein the data are provided as a set of self-

5 contained files distributed over two or more file systems, each file system having a data path, which when executed by a computer, cause the computer to:

- retrieve a virtual file system comprising pointers to the distributed files within the two or more file systems; and

10 - read a set of files from the two or more file systems based on the virtual file system using two or more I/O threads.

According to the invention a virtualized file system layer is created, enforced by either script, daemon and/or Multipath I/O
15 driver, where each file appears in a set of files, e.g. a file sequence. The files are scattered on several file systems, each with its own separate data path. Reading and writing a set of files, e.g. a chronological sequence of files, is thus accomplished by several I/O threads, usually by different data
20 channels. This maximizes the effective overall I/O throughput. Only the availability and I/O-independence of those paths is required.

The proposed solution provides a cost-effective alternative to
25 storage systems that support multipath I/O at global architecture, hardware and/or operating system level, e.g. Storage Area Networks. This is especially advantageous when just high-performance throughput is paramount, while other higher-level features, like shared access or high availability,
30 are not necessary.

Advantageously, a folder created in the virtual file system or moved within the virtual file system is identically created in the two or more file systems or moved within the two or more

file systems. This ensures that all file systems replicate the same folder hierarchy.

Preferably, the distribution of the files over the two or more
5 file systems is based on at least one of file size, file date, metadata associated to the files, or modulo divisions on a file index in the set of files. All these properties are well suited for distributing the files as required.

10 Preferably, when a file comprising a multi-digit sequence in the file name is to be stored in the storage system, matching files already stored in the storage system and belonging to the same set of files are searched for. A storage location for the file to be stored is then determined in dependence of whether
15 or not matching files are found in the storage system. Using such multi-digit sequences in the file name, which are often encountered in professional environments, the various files of the file set can reliably be distributed to the various file systems.

20 In principle, the proposed solution is applicable to any set of files. It is especially advantageous when the files have roughly the same size and need to be accessed as fast as possible. For example, the set of files may be a frame-per-file
25 sequence of video frames, e.g. a video clip. Of course, the video frames may likewise be stereoscopic or multiview video frames. In this case the video frames for different views are preferably stored in respective different file systems.

30 Despite originating from technical problem encountered in high-profile post-production facilities, the proposed solution is not limited to such applications. With the advent of ultra-high-resolution (UHD) consumer devices, throughput requirements are steeply ramping up, even for the non-professional storage

market. For this reason the described approach is immediately applicable to file system management of consumer media systems, including embedded ones like SmartTVs, set-top boxes, video servers or smartphones/tablets capable of high-resolution video streaming on either Direct-Attached Storage or on network and cloud storage.

For a better understanding the invention shall now be explained in more detail in the following description with reference to the figures. It is understood that the invention is not limited to this exemplary embodiment and that specified features can also expediently be combined and/or modified without departing from the scope of the present invention as defined in the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 schematically illustrates a method according to the invention for handling data in a storage system; and

Fig. 2 schematically illustrates an apparatus configured to perform a method according to the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

In the following the invention shall be explained with reference to video frames provided as a frame-per-file sequence. Of course, the proposed solution is applicable to any data provided as a set of files.

As already indicated before, in high-profile post-production for digital intermediates and digital cinema, high-resolution uncompressed video footage is handled. This video footage is traditionally stored as frame sequences, i.e. each subsequent

frame of a time-continuous video clip is stored as a single file. Maximizing the I/O performance when reading and writing video clips based on frame-per-file sequences is crucial.

5 All the files belonging to the same logical clip typically share the file system metadata shown below:

- file path (i.e. the files reside in the same folder, e.g. /path/to/clip);
- 10 • base filename (e.g. basename-);
- file extension (which logically mandates the file-type encoding the frame, e.g. .dpx).

For example, a video clip consisting of 2000 frames may be
15 stored as the following file sequence:

- /path/to/clip/basename-289368.dpx;
- /path/to/clip/basename-289369.dpx;
- /path/to/clip/basename-289370.dpx;
- 20 •
- /path/to/clip/basename-300367.dpx.

Stereoscopic clips are typically represented on file systems using either two separate frame sequences 'L' and 'R', i.e. one
25 for the left eye and one for the right eye, or one interleaved frame progression, where each R-frame immediately follows its synchronous L-frame in the file sequence numbering order.

During real-time operations, i.e. video playback or recording,
30 the access time is tied to the single-I/O- or multi-I/O-threaded capabilities of the volume tied to the specific file system.

Furthermore, most file systems and storage managers on operating system level, particularly if embedded in mobile or non-professional equipment, do not take into account such high-level file system hierarchies when doing the initial allocation of blocks on the data volume representing the file systems.
5 Exceptions are specifically built storage systems, like those used in media-aware Storage Area Networks and Content Delivery Networks. In all other cases, little to none pre-allocation strategies are employed. Therefore, locality of reference is
10 lost across the same frame-per-file sequence, which poses additional threats to the related I/O throughput.

A method according to the invention for handling data provided as a set of files in a storage system is schematically
15 illustrated in in Fig. 1. For storing the data in a storage system, a set of self-contained files is received 10. The files are then distributed 11 over two or more file systems based on one or more properties of the files, where each file system has an independent data path. In order to be able to access the
20 distributed files, a virtual file system is generated 12, which comprises pointers to the distributed files within the two or more file systems. For reading data from the storage system the virtual file system is retrieved 13. A set of files is then read 14 from the two or more file systems based on the virtual
25 file system using two or more I/O threads.

An apparatus 20 configured to implement the method is schematically depicted in Fig. 2. The apparatus 20 comprises an input 21 for receiving data provided as a set of self-contained
30 files and a storage system 25 for storing the data. For storing data, a file distributor 22 distributes 11 the files over two or more file systems belonging to the storage system 25 using a data interface 24. The distribution of the files over the two or more file systems is based on one or more properties of the

files. Each file system has an independent data path. A file system generator 23 generates 12 a virtual file system comprising pointers to the distributed files within the two or more file systems. For reading data, a file system retrieving unit 26 retrieves 13 the virtual file system. The data interface 24 then reads 14 a set of files from the two or more file systems based on the virtual file system using two or more I/O threads. The retrieved data are made available at an output 27. Of course, the different units 22, 23, 26 may likewise be fully or partially combined into a single unit or implemented as software running on a processor. In addition, the input 21 and the output 27 may likewise be combined or partially combined into a single bi-directional interface.

The proposed solution has a considerably increased performance. The approach presumes $N > 1$ different storage volumes, each with its own file system structure on top, and each with its own pre-defined data path. At operating system level, a data path here means a block-level handler, like for example `/dev/*/sdxx` files in Linux, or a similar UNC path `\\.\67a578c67def...` in Windows. Different Physical paths for each data path, e.g. different buses, SCSI hosts, iSCSI targets, or even different network volumes, may be a plus or a drawback, depending on the specific implementation and on the rest of the infrastructure.

An $(N+1)$ th file system is also used, which is not necessarily tied to a real-time-capable, otherwise I/O performing storage system. This file system is the only exposed file system at virtual level, although the device or application actually performing the I/O must physically access all the other N data paths, i.e. silently mount them in its own file system.

For the sake of clarity, let a Linux- or Android-based operating system be considered and `/dev/block/lun1`,

/dev/block/lun2, ..., and /dev/block/lunN be the N volumes mounted over points /mnt/dp1, /mnt/dp2, ..., and /mnt/dpN, respectively. Let the above mentioned $(N+1)$ th file system be considered to be locally rooted in /media/store. User-

5 applications should not have direct visibility of the mount points /mnt/dp k ($1 \leq k \leq N$), but still be able to resolve I/O request at file system level as assigned to folder /media/store. This folder should only contain frame sequences.

10 A script or demon program running on the system takes care of file creation and other POSIX-like requests on either the whole /media/store folder or on selected sub-folders of this folder, including the following rules:

15 • Any folder created or moved within /media/store is in the same way created or moved within any of the file systems /mnt/dp k as well. Consequently, all of them replicate the same folder hierarchy.

• When a file whose name ends with a multi-digit sequence, e.g. 20 representing integer number m , is created in or written to a path, other files belonging to the same sequence, i.e. to the same video clip, are searched for:

o If no matching file sequence is found, this is considered to be the first file and it is actually created on /mnt/dp t , 25 where $t = 1$.

o If a matching sequence is found in the storage system (with first frame t_0) the m th file is actually created in /mnt/dp t , where $t = (m - t_0) \% N$.

o A soft-link for that very file is created pointing from 30 /media/store, under the homologous folder, to the right path under the mount-point /mnt/dp t .

For example, if $N = 3$ and the usual 1000 files need be created under /media/store/clip1/, they would all be soft-linked as follows:

```
/media/store/clip1/frm.289368.exr -> /mnt/dp1/clip1/frm.289368.exr
/media/store/clip1/frm.289369.exr -> /mnt/dp2/clip1/frm.289369.exr
/media/store/clip1/frm.289370.exr -> /mnt/dp3/clip1/frm.289370.exr
/media/store/clip1/frm.289371.exr -> /mnt/dp1/clip1/frm.289371.exr
/media/store/clip1/frm.289372.exr -> /mnt/dp2/clip1/frm.289372.exr
/media/store/clip1/frm.289373.exr -> /mnt/dp3/clip1/frm.289373.exr
/media/store/clip1/frm.289374.exr -> /mnt/dp1/clip1/frm.289374.exr
.....
/media/store/clip1/frm.300366.exr -> /mnt/dp3/clip1/frm.300366.exr
/media/store/clip1/frm.300367.exr -> /mnt/dp1/clip1/frm.300367.exr
```

5

During both sequential write and read operations on the virtual file sequence shown on the left in the above list, the storage allocation scheme becomes less important as the file access times. When an application accesses the video using N different threads, it actually uses all N independent data paths at the same time. This increases the transfer speed, as it is a real parallel I/O, using a scheme similar to Multipath I/O, but at a far higher model level. At the same time implementation of this scheme is less expensive, as it does not require dedicated hardware or firmware. Furthermore, file access times and internal file fragmentation are also less relevant to sequential data reads, as sequential operations on the same local file system occur only once per every N files.

20 For stereoscopic file sequences two different possible implementations are described in the following, where two different frame-per-file sequences are accessed, one for left-eye, or L-frames, and one for right-eye, or R-frames.

If there are $N=2$ independent file systems, each capable of satisfying I/O requirements for a single monoscopic clip with the same specifications, their block-level devices /dev/block/lun1 and /dev/block/lun2 are mounted on /mnt/Lstor and /mnt/Rstor, respectively. In this case the implementation described above is used. However, the background file-managing script takes care of the stereoscopic clips in a slightly different manner:

```

/media/store/clip/frm-L_00000.tif -> /mnt/Lstor/clip/frm_0000.tif
/media/store/clip/frm-R_00000.tif -> /mnt/Rstor/clip/frm_0000.tif
/media/store/clip/frm-L_00001.tif -> /mnt/Lstor/clip/frm_0001.tif
/media/store/clip/frm-R_00001.tif -> /mnt/Rstor/clip/frm_0001.tif
/media/store/clip/frm-L_00002.tif -> /mnt/Lstor/clip/frm_0002.tif
/media/store/clip/frm-R_00002.tif -> /mnt/Rstor/clip/frm_0002.tif
.....
/media/store/clip/frm-L_00999.tif -> /mnt/Lstor/clip/frm_0999.tif
/media/store/clip/frm-L_00999.tif -> /mnt/Rstor/clip/frm_0999.tif

```

Otherwise, the two above approaches can be combined, i.e. a two-level striping using $N > 2$ storage pools, where L- and R-frames are interleaved with each other on consecutive lower level file systems. In both cases, the capability to have at least two I/O threads in the application allows for a simultaneous I/O handling of the two simultaneous eye-frames. Of course, this approach can easily be extended to multiview clips. For each view a dedicated file system is used, i.e. video frames for different views are stored in respective different file systems.

In the above examples the files are distributed based on their file names. Of course, this is not the only possibility to distribute the files. The distribution may likewise be based on other attributes, e.g. file size, file type or metadata available for the files. For example, larger files may be

stored in physical file systems that have more (free) space. Accordingly, smaller files may be stored on smaller or fuller file systems. Similarly, files that are of multimedia type, e.g. video files associated to high bit rates, may be stored on
5 file systems whose data paths have a higher bandwidth in order to speed up the file transfer process, as well as the later playback operations. Files associated to lower bit rates may accordingly be stored on file systems with a lower bandwidth.

CLAIMS

1. A method for storing data in a storage system (25), wherein
the data are provided as a set of self-contained files, the
method **comprising** the steps of:
 - distributing (11) the files over two or more file systems
based on one or more properties of the files, each file
system having an independent data path; and
 - generating (12) a virtual file system comprising pointers
to the distributed files within the two or more file
systems.
2. The method according to claim 1, **wherein** a folder created in
the virtual file system or moved within the virtual file
system is identically created in the two or more file
systems or moved within the two or more file systems.
3. The method according to claim 1 or 2, **wherein** distribution
of the files over the two or more file systems is based on
at least one of file size, file date, metadata associated to
the files, or modulo divisions on a file index in the set of
files.
4. The method according to claim 3, **wherein** when a file
comprising a multi-digit sequence in the file name is to be
stored in the storage system, matching files already stored
in the storage system and belonging to the same set of files
are searched for.
5. The method according to claim 4, **wherein** a storage location
for the file to be stored is determined in dependence of
whether or not matching files are found in the storage
system.

6. The method according to any of the preceding claims, **wherein** the set of files is a frame-per-file sequence of video frames.
- 5 7. The method according to claim 5, **wherein** the video frames are stereoscopic or multiview video frames.
8. The method according to claim 7, **wherein** video frames for different views are stored in respective different file
10 systems.
9. An apparatus (20) configured to store data in a storage system (25), wherein the data are provided as a set of self-contained files, the apparatus (20) **comprising**:
15 - a file distributor (22) configured to distribute (11) the files over two or more file systems based on one or more properties of the files, each file system having an independent data path; and
- a file system generator (23) configured to generate (12) a
20 virtual file system comprising pointers to the distributed files within the two or more file systems.
10. A computer readable storage medium having stored therein instructions enabling storing data in a storage system (25),
25 wherein the data are provided as a set of files, which when executed by a computer, cause the computer to:
- distribute (11) the files over two or more file systems based on one or more properties of the files, each file system having an independent data path; and
30 - generate (12) a virtual file system comprising pointers to the distributed files within the two or more file systems.
11. A method for reading data from a storage system (25), wherein the data are provided as a set of files distributed

over two or more file systems, each file system having an independent data path, the method **comprising** the steps of:

- retrieving (13) a virtual file system comprising pointers to the distributed files within the two or more file

5 systems; and

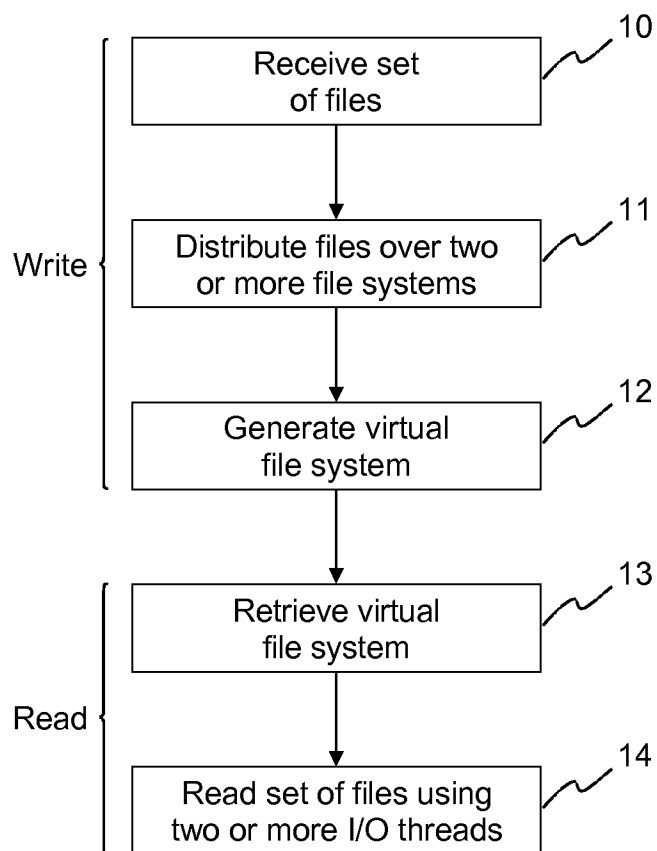
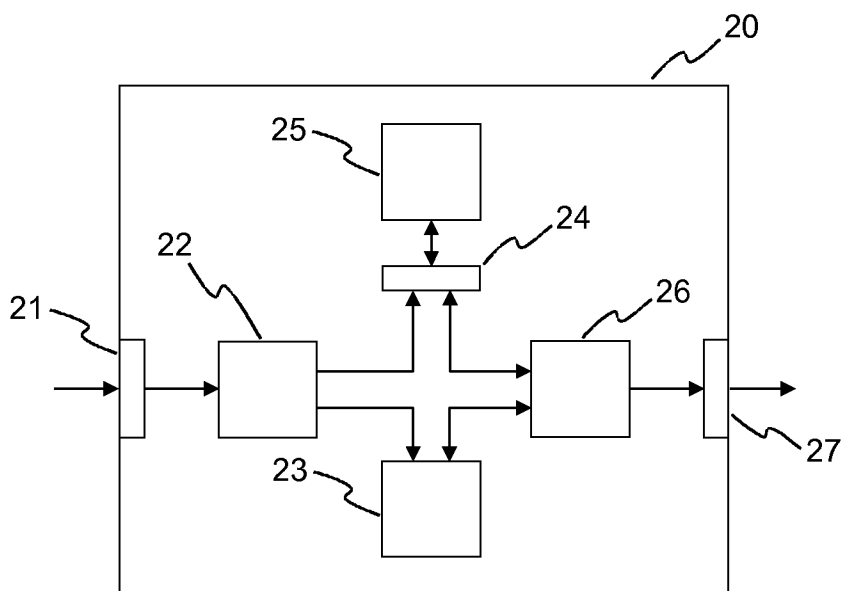
- reading (14) a set of files from the two or more file systems based on the virtual file system using two or more I/O threads.

10 12. An apparatus (20) configured to read data from a storage system (25), wherein the data are provided as a set of self-contained files distributed over two or more file systems, each file system having an independent data path, the apparatus **comprising**:

15 - a file system retrieving unit (26) configured to retrieve (13) a virtual file system comprising pointers to the distributed files within the two or more file systems; and
- a data interface (24) configured to read (14) a set of files from the two or more file systems based on the virtual
20 file system using two or more I/O threads.

13. A computer readable storage medium having stored therein instructions enabling reading data from a storage system (25), wherein the data are provided as a set of self-
25 contained files distributed over two or more file systems, each file system having an independent data path, which when executed by a computer, cause the computer to:

- retrieve (13) a virtual file system comprising pointers to the distributed files within the two or more file systems;
30 and
- read (14) a set of files from the two or more file systems based on the virtual file system using two or more I/O threads.

**Fig. 1****Fig. 2**

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2014/078498

A. CLASSIFICATION OF SUBJECT MATTER
 INV. G06F17/30
 ADD.

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)
 G06F

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 8 515 904 B1 (DWYER III THOMAS J [US] ET AL) 20 August 2013 (2013-08-20)	1-5,9-13
Y	abstract; figures 1-3 column 2, line 36 - column 8, line 53	6-8
Y	----- WO 99/14687 A2 (MICROSOFT CORP [US]) 25 March 1999 (1999-03-25)	6-8
A	abstract; figures 1-3 page 1, line 5 - page 5, line 14	1-5,9-13
A	----- US 2004/199578 A1 (KAPCZYNSKI MARK JOSEPH [US] ET AL) 7 October 2004 (2004-10-07) abstract; figure 2 paragraph [0027] - paragraph [0030] ----- -/-	1-13



Further documents are listed in the continuation of Box C.



See patent family 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 application or patent 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

"T" 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

"&" document member of the same patent family

Date of the actual completion of the international search

16 March 2015

Date of mailing of the international search report

25/03/2015

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
 NL - 2280 HV Rijswijk
 Tel. (+31-70) 340-2040,
 Fax: (+31-70) 340-3016

Authorized officer

Wienold, Norbert

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2014/078498

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2002/078174 A1 (SIM SIEW YONG [US] ET AL) 20 June 2002 (2002-06-20)	1-5,9-13
Y	abstract; figure 7 paragraph [0107] - paragraph [0110] paragraph [0084] - paragraph [0095] -----	6-8

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/EP2014/078498

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 8515904	B1	20-08-2013	US 8515904 B1 20-08-2013
		US 8892507 B1	18-11-2014
WO 9914687	A2	25-03-1999	US 6134596 A 17-10-2000
		WO 9914687 A2	25-03-1999
US 2004199578	A1	07-10-2004	CA 2523780 A1 28-10-2004
		EP 1611507 A1	04-01-2006
		US 2004199578 A1	07-10-2004
		WO 2004092947 A1	28-10-2004
US 2002078174	A1	20-06-2002	AT 516657 T 15-07-2011
		EP 1892921 A2	27-02-2008
		EP 1988684 A2	05-11-2008
		US 2002078174 A1	20-06-2002
		US 2002083118 A1	27-06-2002
		US 2002083187 A1	27-06-2002
		US 2002112069 A1	15-08-2002
		US 2003026254 A1	06-02-2003
		US 2003031176 A1	13-02-2003
		US 2003046369 A1	06-03-2003
		US 2005198238 A1	08-09-2005