(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 November 2015 (19.11.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/175411 A1

(51) International Patent Classification:
G06F 17/30 (2006.01) *H03M 13/00* (2006.01)

(21) International Application Number:
PCT/US2015/030163

(22) International Filing Date:
11 May 2015 (11.05.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/992,286 13 May 2014 (13.05.2014) US
62/053,255 22 September 2014 (22.09.2014) US

(71) Applicant: **CLOUD CROWDING CORP.** [US/US]; 9 East 40th Street-7th Floor, New York, NY 10016 (US).

(72) Inventors: **YANOVSKY, David**; 6 Ino Shaki Street, Jerusalem (IL). **NAMORADZE, Teimuraz**; Narva Mnt 5-100, Tallin (EE).

(74) Agent: **KASDAN, Abraham**; Wiggin And Dana LLP, One Century Tower, New Haven, CT 06508-1832 (US).

(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).

Published:

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

(54) Title: DISTRIBUTED SECURE DATA STORAGE AND TRANSMISSION OF STREAMING MEDIA CONTENT

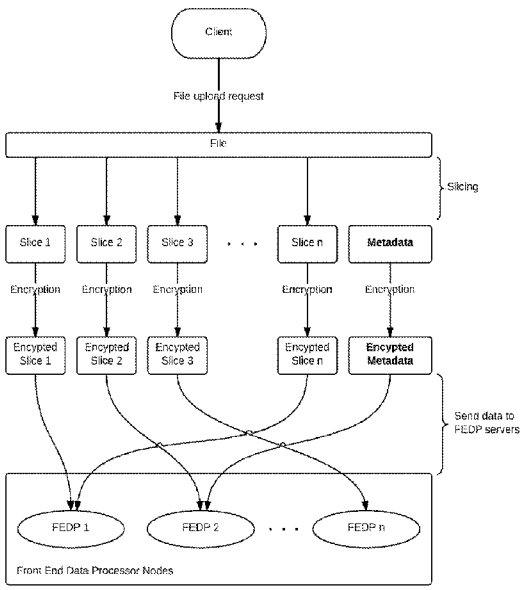


FIG. 4A

(57) Abstract: Disclosed is a method for the distributed storage and distribution of data. Original data is divided into fragments and erasure encoding is performed on it. The divided fragments are dispersedly stored on a plurality of storage mediums, preferably that are geographically remote from one another. When access to the data is requested, the fragments are transmitted through a network and reconstructed into the original data. In certain embodiments, the original data is media content which is steamed to a user from the distributed storage.

WO 2015/175411 A1

Distributed Secure Data Storage and Transmission of Streaming Media Content

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This non-provisional application claims priority to United States Provisional Patent Application No. 61/992,286, entitled "A Method for Data Storage," filed May 13, 2014, and United States Provisional Patent Application No. 62/053,255, entitled "A Method for Media Streaming," filed September 22, 2014. The disclosures of United States Provisional Patent Application Nos. 61/992,286 and 62/053,255 are hereby incorporated by reference herein in their entirety.

FIELD OF THE DISCLOSURE

[0002] The subject matter of the present disclosure generally relates to secure data storage and transmission, and more particularly relates to distributed secure data storage and transmission for use in media streaming and other applications.

BACKGROUND OF THE DISCLOSURE

[0003] The promise of cloud computing to revolutionize the landscape of information technology (IT) infrastructure is based upon the premise that both hardware and software resources previously maintained within a company's own data center or local network can be made available through a network of cloud servers hosted on the Internet by third parties, thereby alleviating the need for companies to own and manage their own elaborate IT infrastructures and data centers. However, in order to convince companies to transition their data storage and computing requirements to such third-party "cloud" server(s), the cloud servers need to provide a level of performance, data security, throughput and usability criteria that will satisfy customers' needs and security concerns. For example, storage resources remain a bottleneck to full scale adoption of cloud computing in the enterprise space. Current cloud-based storage resources can suffer from serious performance concerns, including dangerous security vulnerabilities, uncertainties in availability, and excessive costs. Cloud-based storage, or Storage as a Service (StAAS) must create a virtual "storage device" in the cloud which can compete with current in-house storage capacity found in the enterprise data center.

[0004] Current cloud-based storage solutions are most often based on conventional file storage (CIFS, NFS) technology, in which whole files and groups of files are stored in one physical server location. This approach fails to offer acceptable data transfer rates under typical communications conditions found on the Internet. Latency is poor, and the end-user or consumer perceives a performance wall in even the best designed cloud applications. In addition, transfer of large amounts of data can take an inordinate amount of time, making it impractical. For example, a 1 Tb data transfer through the cloud using current technologies could require weeks to complete.

[0005] Cloud storage, in which complete files are stored in a single location, also provides a tantalizing target for hackers interested in compromising sensitive company information. All the efforts put into design of security procedures in the enterprise data center can vanish with one determined hacker working over the Internet. It is therefore highly desirable to increase the security of cloud-based storage systems.

[0006] Cloud storage solutions are also highly vulnerable to "outages" that may result from disruptions of Internet communications between the enterprise client and its cloud storage server. These outages can be of varying duration, and can be lengthy, for example, in the event of a denial of service (DOS) attack. An enterprise can suffer significant harm if it is forced to cease operations during these outages.

[0007] Cloud storage solutions based on storage of whole files in one server location also make disaster recovery a potential pitfall if the server location is compromised. If replication and backup are also handled in the same physical server location, the problem of failure and disaster recovery could pose a real danger of massive data loss to the enterprise.

[0008] Current technology cloud storage solutions require the storage overhead of complete replication and backup to ensure the safety of the stored enterprise data. In typical current cloud storage technology setups this can require up to 800% redundancy in stored data. This large amount of required data redundancy adds a tremendous overhead in costs to maintain the storage capacity in the cloud. The need for such redundancy not only increases cost, but also introduces new problems for data security. In addition, all

this redundancy also brings with it performance decreases as cloud servers use replication constantly in all server data transactions.

[0009] As Internet connections have improved in their ability to handle high throughputs of data, media streaming has become a very popular way to provide media content, such as videos and music, in a way that reduces the risk of unscrupulous copying. Cloud storage plays an important role in many media content streaming schemes. Typically, the media content resides on a company's web server. When requested by a user, the media content is streamed over the Internet in a steady stream of successive data segments that are received by the client in time to display the next segment of the media file, resulting in what appears to be seamless playback of the audio or video to the user.

[0010] Currently, media streaming technology is based upon the concept of transferring media files through web servers, in compressed form, as a segmented stream of data which is received by the client in time to play the next segment of the media file so as to provide continuous playback. In some cases, the rate of data transfer exceeds the rate at which the data is played, and the extra data is buffered for future use. If the rate of data transfer is slower than the rate of data playback, the presentation will stop while the client collects the data needed to play the next segment of the media. The advantages of streaming media technology are found in the fact that the client does not need to wait to download an entire large media file (e.g., a full length movie) and the fact that the on-demand download nature lends itself to process digital rights management (DRM) schemes that protect against unauthorized copying of the media content by the client.

[0011] Current media streaming technology stores a complete copy of the entire media file on a web or media server to which the client connects to receive the stream of data. Data losses during the transmission process can easily interrupt the transfer process and halt the playback of the media content on the client. To avoid such problems, the prior art technology often will place the same media file on multiple server nodes, and multiple data centers throughout the world, whether they be public or private, so the user can connect to a server node near them. While this is necessary to insure the steady data transfer rates needed in the face of data packet loss due to connectivity issues, deploying multiple copies of the same file on many servers throughout the world places a major

burden on streaming media providers.

[0012] The subject matter of the present disclosure is directed to mitigating and/or overcoming one or more of the problems set forth above and to providing for a more secure data storage and transmission method, and more particularly to providing for a more secure data storage and transmission method for use in media streaming and other applications.

BRIEF SUMMARY OF THE DISCLOSURE

[0013] Disclosed is a method and system for secure distributed data storage that is particularly suited to the needs of streaming media.

[0014] A particular data storage embodiment involves separating a media data file into multiple discrete pieces, erasure coding these discrete pieces, and dispersing those pieces among multiple storage units, wherein no one storage unit has sufficient data to reconstruct the data file. A map is generated, showing in which storage units each of the discrete pieces of the data file is stored. In particular, a unique identifier is assigned to each discrete piece and a map of the unique identifiers is used to facilitate the reassembly of the data files.

[0015] In another embodiment, the data storage technique disclosed herein involves separating a data file into slices, assigning a unique identifier to each slice, creating a map of the unique identifiers to facilitate reassembly, fragmenting of each slice into discrete slice fragments, erasure coding of the slice fragments, dispersing the fragments among multiple storage units wherein no storage unit has sufficient data to reconstruct the data file, and generating a map of which storage units house what fragments.

[0016] The goals of both data security and packet loss mitigation are remedied by the disclosed erasure coding process. First, data is coded into unrecognizable pieces, during the erasure coding process thereby providing a high degree of security. Second, the erasure coded data provides for error correction in the event a data loss. While erasure

coding increases the amount of data, data losses that are less than the increase in data size can be accommodated, and recovered. Notably, the processed and erasure-coded data that is stored in accordance with preferred embodiments does not include any replications of the original data, thus strongly increasing security.

[0018] In one embodiment, a method for storing streaming media content includes separating a digital media content file into discrete pieces or fragments, erasure coding the discrete pieces and dispersing the discrete pieces among multiple storage units, wherein no one storage unit has sufficient data to reconstruct the media content. In a preferred embodiment, a map is generated that details in which storage unit each of the discrete pieces is stored. Unique identifiers are assigned to each discrete piece of the media content and a map of the unique identifiers is used to facilitate reassembly of the media content. For example, the map can be used by a client device to reconstruct the media file and allow playing of the media content on the client device, either in a browser or otherwise.

[0019] In another embodiment, a method of data storage includes the steps of separating a data file into slices, assigning unique identifiers to each slice, creating a map of the unique identifiers, fragmenting the slices into discrete pieces or fragments, erasure coding the discrete pieces, dispersing the discrete pieces among multiple storage units, wherein no storage unit has sufficient data to reconstruct the data file, and, generating a map showing in which storage units each of the discrete pieces is stored. Decoding is performed on a client device by using the maps to allow playback and/or further storage of a streamed media file.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] The foregoing summary, preferred embodiments, and other aspects of the present disclosure will be best understood with reference to the following detailed description of specific embodiments, when read in conjunction with the accompanying drawings, in which:

[00021] Figure 1 is a schematic diagram of three layers of an exemplary storage system.

[0022] Figure 2 is a diagram showing the various stages of file processing according to an exemplary embodiment.

[0023] Figure 3 is a chart outlining various steps undertaken during file processing according to an exemplary embodiment.

[0024] Figure 4A is a diagram of a first section of file processing according to an exemplary embodiment.

[0025] Figure 4B is a diagram of the erasure coding of file slices to produce slice fragments for dispersal according to an exemplary embodiment.

[0026] Figure 5 is a detailed diagram of the upload process of a file to data storage nodes according to an exemplary embodiment.

[0027] Figure 6 is a chart of the various detailed steps undertaken during a download process of data from data storage to a client, according to an exemplary embodiment.

[0028] Figure 7A is a diagram of a client download request being made to the CSP, according to an exemplary embodiment.

[0029] Figure 7B is a diagram of a request for slice fragments according to an exemplary embodiment.

[0030] Figure 8 is a detailed diagram of the interaction between the CSP, FEDP and SNN during a file download process.

[0031] Figure 9 is a diagram of a data garbage collection process according to an embodiment.

[0032] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0033] Disclosed, herein, is a cloud storage technology for streaming media files, which breaks up each data file into file slice fragments which are stored on a series of cloud servers, that are preferably dispersed among different geographical locations. In an embodiment, client enterprise media data is disassembled into file slice fragments using object storage technology. All the resulting file slice fragments are encrypted, and optimized for error correction using erasure coding, before dispersal to the series of cloud servers. This creates a virtual "data device" in the cloud. The servers used for data storage in the cloud can be selected by the client to optimize for both speed of data throughput and data security and reliability. For retrieval, the encrypted and dispersed file slice fragments are retrieved and rebuilt into the original file at the client's request. This dispersal approach creates a "virtual hard drive" device in which a media file is not stored in a single physical device, but is spread out among a series of physical devices in the cloud which each only contain encrypted "fragments" of the file. Access of the file for the purposes of moving, deleting, reading or editing the file is accomplished by reassembling the file fragments rapidly in real time. This approach provides numerous improvements in speed of data transfer and access, data security and data availability. It can also make use of existing hardware and software infrastructure and offers substantial cost reductions in the field of storage technology.

[0034] While the dispersed storage of data, including in particular streaming media data, on cloud servers is one particularly useful application, the same technology is applicable to configurations in which the data may be stored on multiple storage devices which may be connected by any possible communications technology such as LAN's or WAN's. The speed and security benefits of the disclosed technology could remain within the devices of an information technology (IT) data center, where the final storage devices are multiple physical hard disks or multiple virtual hard disks. An IT user may choose to use all the storage devices available throughout the company which are connected by a high speed LAN in which the disclosure's technology is implemented. The multiple storage devices may even be spread across multiple individual users in cyberspace, with

files stored on multiple physical or virtual hard disks which are available in the network. In each case, the speed of data transfer and security of data storage in the system are greatly enhanced.

[0035] Uses for the disclosed subject matter include secondary data storage, for backup or disaster recovery purposes. The disclosed subject matter is also applicable to primary storage needs where the files are accessed without server-side processing. In certain embodiments, this includes storage of media content, including without limitation video or audio content that can be made available for streaming through the Internet.

[0036] Data Storage Advantages

[0037] The disclosed storage technology presents numerous advantages over existing systems. Among these advantages are the following:

[0038] A. Data Transfer Rates

[0039] Compared to existing cloud storage technology, the disclosed embodiments permit substantial improvements in the speed of data transfer under typical Internet communication conditions. Speeds of up to 300 mbps have been demonstrated, which would mean for example, that transfer of a 1 Tb file, which could take a month using some existing systems, can be completed in 10 hours. This speed improvement stems from several factors.

[0040] When reconstructing a file its attendant "pieces" are transferred from/to multiple servers in parallel, resulting in substantial throughput improvements. This can be likened to some of the popular download accelerator technologies in use today, which also open multiple channels to download pieces of a file, resulting in substantial boost in download rates. Latency bottlenecks that might occur in one of the transfer connections to one of the cloud servers do not stop the speedier transfers to the other servers which are operating under conditions of normal latency.

[0041] The inherent improvements in data security and reliability stemming from distributed storage eliminates the need for constant mirroring of data read/writes through replication, resulting in further speed improvements to throughput.

[0042] Typically, the most resource intensive processing of the data occurs at the server side on one or more very high performance servers in the cloud, which are optimized for speed and connectivity to both the cloud server storage sites and the client sites.

[0043] In particular, erasure coding in certain embodiments is performed at the server side, for example, as described further herein, on multiple data processing servers. These servers may be chosen to have high processing performance, since the erasure coding process is typically a central processing unit (CPU) intensive task. This results in improved performance as compared to erasure coding done at the client side, which may lack the hardware and software infrastructure to efficiently perform erasure coding, or on a single server. Moving such processing to an optimized group of servers decreases the load and performance requirements at the client side, compared to existing designs.

[0044] B. Data security

[0045] The disclosed "virtual device" storage offers significant improvements in terms of data security over previous designs. By breaking up each media file into many file slice fragments and dispersing the file slice fragments over many cloud storage locations, preferably at geographically dispersed locations, a hacker would find it extremely difficult to reassemble the file into its original form. In addition, the file slice fragments are all encrypted in certain embodiments, adding another layer of data security to confound a would-be hacker. A successful hack into one of the cloud storage locations will not give the hacker the ability to reassemble the full media file. This is a significant improvement in data security over previous designs.

[0046] In certain embodiments, the servers used for both processing and storage of file slice fragments may be shared by multiple clients, with no way for a hacker to identify from the data slices to which client they may belong. This makes it even more difficult for

a hacker to compromise the security of file data stored using this technology. File slice fragments may be dispersed randomly to different cloud storage servers, further enhancing the security of the data storage. In certain embodiments, not even the client may know exactly the locations to which all the file slice fragments have been directly dispersed. Also, there is no one place where all the keys are stored to reassemble the file slice fragments and/or decrypt the file slice fragments. Lastly, as an additional enhancement to data security, a two dimensional model of metadata storage may be used, in which metadata needed to reconstruct the data is stored on both the client side and on remote cloud storage servers.

[0047] C. Data Availability

[0048] The disclosed "virtual device" storage also offers improvements in the availability of the data, compared to prior art storage technology. By splitting the file into multiple file slice fragments which are stored on a number of different cloud servers, communications problems between the client location and one of the physical cloud locations may be compensated by normal communications with and low latency at other data locations. The overall effect of having file fragments dispersed among multiple locations is to insulate the overall system from outages due to communications disruptions at one of the sites.

[0049] Preferably, the intermediate server processing nodes discussed below are all comprised of high performance processors and have low latencies. This results in high availability to the client for data transfers.

[0050] Preferably, the intermediate server processing nodes may be chosen dynamically in response to each client request to minimize latency with the client who requests their services. The client may also select from a list of cloud storage servers to be used to store the file slice fragments, and can optimize this list based on his geographical location, and the availability of these servers. This further maximizes data availability for each client at the time of each transfer request.

[0051] D. Data reliability

The disclosed "virtual device" storage also provides improvements over the prior art in the reliability of a cloud data storage system. Separation of each file into file slice fragments means that hardware or software failures, or errors at one of the physical cloud storage locations will not prevent access to the file, as would be the case if the entire file is stored in one physical location, as in certain previously existing systems. Further, the use of the erasure coding technology discussed herein insures high quality error correction capabilities in the system, enhancing both data security as well as reliability. The combination of file slice fragments and the erasure coding techniques used herein provides major advances to reliability to encourage enterprise adoption of cloud technology.

[0052] E. Use of existing cloud infrastructure resources

[0053] Elements of the disclosed subject matter may make use of existing cloud server infrastructures, with both public and private resources. Current cloud providers can be setup with their existing hardware and software infrastructure for use with the disclosed methodology. Most of the enhancements offered by the technology disclosed herein may therefore be available with minimal investment, as currently existing cloud resources can be used either without modification or with minimal modification.

[0054] F. Reduction of infrastructure cost

[0055] Certain embodiments require far less redundancy compared to existing cloud storage technology solutions. As mentioned above, previous storage systems can require as much as 500% additional storage devoted to mirroring and replication. The embodiments disclosed herein may operate successfully with only a 30% redundancy over the original file size because of their higher inherent reliability. Even with only 30% redundancy, higher levels of reliability over existing systems can be achieved. The reduced necessity for high redundancy results in lower costs for cloud storage capacity. With the exponential growth in enterprise data and storage needs seen year to year, this reduction of redundancy is an important factor in making a cloud solution economically viable for an enterprise as a complete replacement for its local data center.

[0056] As further disclosed herein, embodiments of the disclosed "virtual device" storage technology accomplish certain tasks: splitting of files into file slices and file slice fragments which will eventually be transferred to a predetermined number of cloud storage locations; creating maps of the file slices and file slice fragments which describe how the files were split, and at which cloud location a group of file slice fragments are stored, to allow for re-assembly of the file by the client; encrypting the file slices and file slice fragments to provide additional data security; adding erasure coding information to the pieces for error checking and recovery; and garbage collection of orphaned file slice fragments which were not properly written and disassembled or read and reassembled.

[0057] As illustrated in FIG. 1, the basic structure of an exemplary system embodiment may be visualized as including three layers. A first layer is the client-side processor (CSP) which may be located at the client's back office or data center. A client application (such as a web app running in a browser) may be used to access the CSP to both set application parameters and initiate uploads of files from the client's data center to the storage node network and downloads of files from the storage node network to the client's data center. In the Figures, "Slice" is generally used to refer to a file slice, and "atom" is generally used to refer a file slice fragment.

[0058] A second layer of the exemplary system includes front-end data processor (FEDP) which perform intermediate data processing. THE FEDP may be located at multiple dispersed locations in the cloud. Multiple FEDP servers may be available to each client, with each FEDP server providing high processing performance, and high availability connections to the client's location.

[0059] A third layer of an exemplary system embodiment is the storage nodes network (SNN). The SNN may include various cloud storage centers that may be operated by commercial cloud resource providers. The number and identity of the storage nodes in the SNN may be optionally selected by the client using his client application to optimize the latency and security of the storage configuration by choosing storage nodes that exhibit the best average latency and availability from the client's location.

[0060] Figure 1 is a schematic diagram showing the interrelationships between the CSP, FEDP and SNN.

[0061] The basic functions performed by these three layers can be described as follows. The CSP can receive and initiate a request for upload of a file to the SNN from a client app. As a first step, it splits the file into a number of slices, each of a given size. The number and size of the slices may be varied via parameters available to the client app. Each slice may be encrypted with a client key, and assigned a unique identifier. The CSP will also produce a metadata file which maps the slices to allow for their reassembly into the original complete file. This metadata file may be stored at the client's data center and may also be encrypted and copied into the SNN. In an exemplary embodiment, the CSP may then send out the sliced files to the next layer, the front end data processor (FEDP), for further processing.

[0062] The FEDP may receive sliced files from the CSP and further process each slice. This processing may divide each slice into a series of file slice fragments. Erasure coding is performed to provide error correction, for example, in the event some data is lost during the transmission process. The erasure coding, as will be further described herein, will increase the size of each file slice fragment, to provide for error correction. The FEDP may also encrypt the file slice fragment using its own encryption key. The FEDP will create another metadata file which maps all of the file slice fragments back to their original slices, and records which storage nodes network (SNN) servers are to be used to store which file slice fragments. Once, this intermediate processing is performed, the FEDP sends groups of file slice fragments to their designated SNN servers in the cloud, and sends a copy of the metadata file it created to each SNN server.

[0063] At the third layer, the SNN servers will now host the processed file slice fragments in the cloud at normally available cloud hosting servers, waiting to receive a future request through the system for file download. The download process basically reverses the steps described above in the three processing layers, so as to reconstruct the original file or file slices at the CSP.

[0064] Figure 2 illustrates the various stages of file processing discussed above

for each of the CSP, FEDP and SNN during upload of a file to the SNN according to an exemplary embodiment. Figure 3 is a chart of the detailed steps that may be included in a file upload process performed in accordance with an exemplary embodiment.

[0065] File Uploading

[0066] Figures 4A and 4B respectively show the two basic processing stages during the upload process of a file from the CSP to the FEDP and then to the SNN: processing at the CSP of a file into file slices, and processing at the FEDP of file slices to create file slice fragments for dispersal to the SNN's. Figure 5 is another illustration of the upload process in step-by-step fashion, showing some of the intermediate steps.

[0067] File Downloading

[0068] The process of downloading a file which has been previously uploaded to the SNN involves a reversal of the steps used in the upload process. The slice fragments which are stored across many SNN's must be reassembled into file slices using a second metadata file which maps how slice fragments are reassembled into slices. This is done by the FEDP. The file slices so generated must be reassembled by the CSP into a complete file using the first metadata file which maps how the slices are reassembled into a whole file for delivery to the client's data center. The second metadata file is stored redundantly on each of the SNN's used to store the file, and the first metadata file is stored in the client's datacenter and on each SNN as well.

[0069] Figure 6 is a chart of the detailed steps that may be involved in the download process.

[0070] Figure 7A shows the download process among the three layers, showing the requests made between the CSP and the FEDP, and the requests between the FEDP and the SNN. Figure 7B illustrates the steps involved when the FEDP requests slice fragments from the SNN to reassemble a requested file slice using the second metadata file.

[0071] Figure 8 illustrates the detailed steps of the interaction between CSP, FEDP and SNN during the download process.

[0072] Technology Optimizations

[0073] As discussed above, the disclosed method and system provides major improvements in both data throughput, data availability, data reliability and data security.

[0074] The multiple number of upload and download nodes used in the system will speed up both uploading and downloading. A further increase in throughput speed may be obtained by optimizing the latency between the CSP and the FEDP's, and choosing the FEDP's with the best current latency available. There is no need to optimize for latency between the FEDP's and the SNN's, as the FEDP's are set up as high performance, high availability servers which are designed to automatically minimize latency to the SNN's. The use of multiple nodes also decreases the performance hit seen if one particular server path is suffering from high latency.

[0075] The use of many storage nodes for storing file slice fragments greatly increases the security available in the storage of client data. The task of a hacker finding the necessary information to tap into all the disparate slice fragments at a large number of SNN's, and reassemble them into a usable file is very formidable.

[0076] The use of erasure coding for the dispersal of the slice fragments adds an extra layer of reliability through its inherent error checking/correction which allows the system to dispense with the need for multiple data replication, with it's inherent performance hits and security risks.

[0077] Additional Issues

[0078] One area, which remains very resource intensive, as mentioned before, is the erasure coding process, which is very CPU intensive. To address this issue, very high performance FEDP hardware insures that the CPUs (or virtual CPUs) used in these FEDP servers meet the performance needs of the system. In addition, the entire software

package may be coded in "Go" language, including the FEDP servers. The native code objects generated by the "Go" language help to improve overall system performance, particularly in the FEDP servers, where erasure coding takes major CPU resources.

[0079] The client app may be any client agent capable of running on the client's operating system (OS) platforms. Optionally, a client app may be written in Javascript to run in browsers. This helps in making such client app available across a wide variety of physical devices.

[0080] The data storage techniques described above may be designed to use virtualized servers throughout. For example, 3 virtual servers in parallel could be used instead of one real hardware server to improve performance, and insure hardware independence. The current system is based on object storage technology, which treats the data as a mass to be referenced, independent of any particular file structure. The goal was to create a system, which can be transferred into block storage, to suit the current virtualization standards in data storage. The current object model can be easily mapped into block storage in the future.

[0081] In certain embodiments, error correction by way of erasure coding is done on the FEDP, using Reed-Solomon coding. A garbage collection system is also employed at the FEDP, in the event of incomplete reads and writes of the FEDP to/from the SNN's.

[0082] Figure 9 illustrates the steps of the garbage collection process, which is necessary to delete objects which were stored into storage nodes incompletely, i.e. objects for which mask cardinality is less then k. Such objects may rarely appear in the system if for some reason more than n – k data blocks failed to upload and an application terminated unexpectedly. The flow consists of four steps:

1. List Incomplete: Every fixed period of time (which may be a configurable value) retrieve a list of incomplete objects using LIST_INCOMPLETE function of metadata storage.
2. Retrieve UIDs: Retrieve corresponding data blocks UIDs using GET function (see Table 2).
3. Delete Data: Extract storage nodes IDs and data blocks IDs from these UIDs and

delete corresponding data blocks from storage nodes using DELETE function (see Table 1)

4. Delete Metadata: Remove deleted object record from metadata storage using DELETE function

[0083] Applications

[0084] Migration of Enterprise Data from Company Data Centers into the Cloud

[0085] The greatly enhanced data transfer speed, security, reliability and availability of the disclosed technology allows an enterprise to migrate much of its data, including in particular its streaming media content out of their company data centers into the cloud. This will make the company's data available to a far wider range of data consumers both inside and outside the company.

[0086] The disclosed technology permits data storage resources throughout the enterprise which are currently under-utilized will then become available for use as secure storage nodes. This can greatly reduce enterprise storage costs, and allow secure distributed storage networks to proliferate throughout the data structure.

[0087] Ultimately, this same use of under-utilized data storage resources can find its way into the general population of computer owners with their collections of under-utilized storage devices. Vast distributed storage networks can be assembled which will take the older concept behind BitTorrent and supercharge it by adding vastly improved speed and security. The entire mobile device revolution in computer technology is predicated on the availability of data in the cloud. In previous systems, this need has been a weak link in these interlinked technologies, due to the lack of speed and security in cloud storage resources. This is particularly needed now that more private and enterprise clients are accessing data through mobile devices, in particular for streaming media applications. With the face of computer usage headed toward heavy use of mobile devices at the expense of desktops and less mobile laptops, the availability of data to users requires extensive migration of data into the cloud. The disclosed technology aids in making this migration possible.

[0088] Digital media streaming

[0089] The disclosed technology is a natural fit with the needs of digital media streaming technology. The disclosed improvements in speed and security, and greater utilization of available storage resources enables higher streaming rates using today's communications protocols and technologies. The vast amount of storage space required for storage of video, audio and other metadata can further benefit from increased availability and utilization of existing resources and infrastructure, in accordance with the exemplary embodiments disclosed herein.

[0090] Satellite TV

[0091] The large hard drives built into satellite TV technology provide an example of how an under-utilized storage resource can be adapted to use the disclosed technology to establish a fast, secure distributed storage network among the general public of satellite TV users. This resource can greatly enhance the value of the satellite TV network, and open up entirely new commercial opportunities.

[0092] In certain embodiments according to the present disclosure, a highly secure erasure coding algorithm is used to code file fragments to provide for data recovery in case some data is lost due to errors in the transmission process.

[0093] In particular, a Data Mixer Algorithm (DMA) is employed that encodes an object F of size $L = |F|$ into n unrecognizable pieces $F_1, F_2, \ldots F_n$, each of size L/m ($m < n$), so that the original object F can be reconstructed from any m pieces. The core of the DMA is an m-of-n mixer code. Data in the fragments processed with the DMA is confidential, meaning that no data in the original object F can be reconstructed explicitly from fewer than m pieces. An exemplary embodiment of the detailed operation of the DMA will now be described.

[0094] The m-of-n mixer code is a forward error correcting code (FEC), whose output does not contain any input symbols and which transforms a message of m symbols

into a longer message of n symbols, such that the original message can be recovered from a subset of the n symbols of length m.

[0095] The original object F is firstly divided into m segments $S_1, S_2, \ldots S_m$, each of size L/m. Then, the m segments are encoded into n unrecognizable pieces $F_1, F_2, \ldots F_n$ using a m-of-n mixer code, e.g.:

$$(S_1, S_2, \ldots S_m) \cdot G_{m\times n} = (F_1, F_2, \ldots F_n),$$

where $G_{m\times n}$ is a generator matrix of the mixer code and meets the following conditions:

1) Any column of $G_{m\times n}$ is not equal to any column of an $m \times m$ identity matrix
2) Any m columns of $G_{m\times n}$ form an $m \times m$ nonsingular matrix
3) Any square submatrix of its generator matrix $G_{m\times n}$ is nonsingular

The first condition ensures that the coding results in n unrecognizable pieces. The second condition ensures that the original object F can be reconstructed from any m pieces where $m < n$ and the third condition ensures that the DMA has strong confidentiality.

[0096] An effective way to construct a DMA with strong confidentiality from an arbitrary m-of-$(m + n)$ mixer code is:

1) Choose an arbitrary m-of-$(m + n)$ mixer code, whose generator matrix is $G_{m\times(m+n)} = (C_{m\times m}|D_{m\times n})$
2) Construct a DMA that adopts an m-of-n mixer code whose generator matrix is $C_{m\times m}^{-1} \cdot D_{m\times n}$

[0097] For example, the generator matrix may be a Cauchy matrix shown below.

[0098] Any square submatrix of a Cauchy matrix,

$$G_C = \begin{pmatrix} \frac{1}{x_1+y_1} & \frac{1}{x_1+y_2} & \cdots & \frac{1}{x_1+y_m} \\ \frac{1}{x_2+y_1} & \frac{1}{x_2+y_2} & \cdots & \frac{1}{x_2+y_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_n+y_1} & \frac{1}{x_n+y_2} & \cdots & \frac{1}{x_n+y_m} \end{pmatrix},$$

where $x_1,\ldots,x_n,y_1,\ldots,y_n \in Z_p$, $x_i + y_j \neq 0; i \neq j \Rightarrow x_i \neq x_j$ and $y_i \neq y_j$ is nonsingular. Thus, a mixer code based on this matrix has strong confidentiality.

[0099] As another example, the generator code can be a Vandermonde matrix.

[0100] To construct a DMA with strong confidentiality from a mixer code whose generator matrix is a Vandermonde matrix, choose a m-of-$(m + n)$ mixer code with generator matrix

$$G_V = \begin{pmatrix} a_1^0 & a_2^0 & \cdots & a_{m+n}^0 \\ a_1^1 & a_2^1 & \cdots & a_{m+n}^1 \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{m-1} & a_2^{m-1} & \cdots & a_{m+n}^{m-1} \end{pmatrix},$$

where $a_1, a_2, \ldots a_{m+n}$ are distinct.

Then, a DMA with strong confidentiality can be reconstructed, in which the corresponding generator matrix is

$$G_{IDMA} = \begin{pmatrix} a_1^0 & a_2^0 & \cdots & a_m^0 \\ a_1^1 & a_2^1 & \cdots & a_m^1 \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{m-1} & a_2^{m-1} & \cdots & a_m^{m-1} \end{pmatrix}^{-1} \times \begin{pmatrix} a_{m+1}^0 & a_{m+2}^0 & \cdots & a_{m+n}^0 \\ a_{m+1}^1 & a_{m+2}^1 & \cdots & a_{m+n}^1 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m+1}^{m-1} & a_{m+2}^{m-1} & \cdots & a_{m+n}^{m-1} \end{pmatrix}$$

[00101] **Encoding Example**

[0102] Assume we have an object F of size $L = |F|$. In the example, L = 1 048 576 (1Mb file). To encode it the following steps are performed:

1. Chose m and n (see description above). For example, $m = 4$, $n = 6$.
2. Chose a word size w (usually 8, 16, 32, which in this example it will be 8). All the arithmetic will be performed over $GF(2^w)$.
3. Chose a packet size z (must be a multiple of computer's word size, and in this example it will be 256).
4. Calculate coding block size $Z=w \cdot z$, which should also be multiple of m. In this example $Z=8 \cdot 256 = 2048$ (bytes) and it is multiple of 4.
5. Pad original object F with random bytes, increasing it size from L to L' so that L' is multiple of Z.
6. Split object F into pieces of size Z. All following steps will be performed over these pieces, however we will denote them still by F.

7. Segment F into sequences $F = (b_1, \ldots b_m,), (b_{m+1}, \ldots b_{2m}), \ldots$where b_i is a w bits length character. In this example it's just a byte. Denote $S_1 = (b_1, \ldots b_m,)$, etc. for convenience.
8. Apply the mixing scheme:

$$F_i = c_{i1}, c_{i2}, \ldots, c_{in},$$

where

$$c_{ik} = a_i \cdot S_k = a_{i1} \cdot b_{(k-1)m+1} + \ldots + a_{im} \cdot b_{km},$$

where a_{ij} are elements of the $n \times m$ Cauchy matrix (see above)

Note, that size of F_i is $L_i = L/m$, in our example this is 250kb (162 144 bytes)

[0103] Decoding Example

[0104] Assume now, we have m object pieces F_i of size L_i. In our example, $i = 1, 3, 5, 6$, on the assumption that F_2 and F_4 have been lost due to transmission errors. To decode and reconstruct original object F, we perform the following steps:

1. Construct $m \times m$ matrix A from the $n \times m$ Cauchy matrix used for encoding by removing all the rows except rows with numbers i. In our example rows 2 and 4 are removed.
2. Invert the matrix A, and apply de-mixing scheme:

$$\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = A^{-1} \cdot \begin{bmatrix} c_{11} \\ \vdots \\ c_{m1} \end{bmatrix}$$

for each segment $S_1 = (b_1, \ldots b_m,)$, etc.

3. Join segment S_i into original Z-length piece F.
4. Join Z-length blocks together to form original, padded object F.
5. Remove padding from F, making it fit size L.

[0105] In exemplary embodiments, the foregoing methodologies of processing data for distributed storage and erasure encoding that makes the original data unrecognizable, are used to process streaming media content. As explained above, the media file of a content provider is broken up into small file slice fragments in a two-step process. The first step breaks up the whole file (which may be compressed or not

compressed) into a series of file slices. These file slices may be encrypted, and a meta-data file is created which maps how to assembly the slices into the original file.

[0106] The second step takes each file slice and breaks it down into smaller data fragments that are erasure coded in accordance with the foregoing techniques to make the original data unrecognizable. The erasure coding may be performed by a set of high-performance file servers with each separate server conducting erasure coding on its file slice(s). This represents a system of virtual erasure coding distributed across n erasure coding server units. The erasure coding adds a pre-defined level of redundancy to the data collection while creating a series of file slice fragments which are then dispersed to a series of file fragment storage nodes. Optimal redundancy of 30% or higher is desired for the erasure coding used in this process. If the media file is frequently accessed, the system can increase file object redundancy of particular slices.

[0107] The erasure coding technique disclosed herein adds a powerful system of automatic error correction which insures that the client receives the correct data packets for the streamed media file, in spite of packet losses. Each data fragment may also be encrypted in the process of erasure coding. A second meta-data file maps the process needed to re-assemble the file slice fragments into the correct streamed media packets. Typically, a minimum of 5 nodes may be needed to successfully process the data for streaming (although the number of nodes is a function of system loading and other parameters). These nodes do not need to be all located near the client who will be receiving the streamed data, but may be located over a wide geographic service area.

[0108] To playback the streaming media content, clients download from the server nodes the required data fragments which are then re-assembled in the proper order. The reassembly reverses the process by which the data fragments were created. Data fragments are reassembled into file slices, and file slices are then reassembled into at least portions of the original media file. As in all streaming technology, the rate of download and processing of the data fragments should be fast enough to allow on time processing of the data packet currently needed for playing the media. The client application, which may include any device capable of playing streamed media, retrieves the file slice fragments in the proper order to begin playing the streamed media file.

[0109] With streamed media, it is essential that all the data fragments are re-assembled sequentially in the proper order, to view or listen to the media from beginning to end. The client device re-assembles the data fragments by using map data from the meta-data files to properly obtain the fragments in their proper sequence. As with current streaming technologies, if the rate of download is faster than the time needed to display the next packets of media data, the reader will download and assemble future time fragments which are stored in a buffer for use when the media player reaches that time segment. The file fragments may not be actually ever assembled into the original media file, but merely played at the proper time, and stored as data fragments. This increases the security of the digital media being played, if the user does not have legal rights to the media file. Of course, if the user does have legal rights to the original media file, the fragments can be assembled on the client's device in the form of the complete original media file, once all the fragments have been downloaded. Because the media file is transmitted from multiple nodes, the file download rates will far exceed the typical rates seen in prior art technology. Preferably, nodes which have at the moment the best connectivity to the client for downloading of data fragments are employed. Since the data on the nodes is redundant, the client software when reading the streamed data may preferentially choose those nodes with the highest rates of data transfer for use in the download.

[0110] This technology is applicable to all types of client devices: desktops, laptops, tablets, smartphones, etc. It does not have to replace the current streaming technology software, but can merely add another layer on top of it for using map files to reassemble the required data fragments in the proper order.

[0111] Advantages Over Previous Systems

[0112] The disclosed distributed storage and erasure coding-based streaming technology offers substantial improvements over the limitations discussed above in prior art streaming technologies.

[0113] A. Speed of data transfer

[0114] For the reasons discussed above, the disclosed embodiments offer substantial improvements in speed of data transfer over typical internet communication conditions compared to prior art streaming technology.

[0115] While a media content provider may choose to disperse the data fragments to high performance servers in the cloud, he may also choose to store the data fragments on multiple storage devices connected in any other type of network. When reconstructing the media file the "pieces" may be transferred from/to multiple servers in parallel, resulting in substantial throughput improvements. This can be likened to the popular download accelerator technologies in use today which also open multiple channels to download pieces of a file, resulting in substantial boost in download rates. Latency bottlenecks in one of the transfer connections to one of the node servers will not stop the speedier transfers to the other servers which are operating under conditions of normal latency. The higher speed of data transfer allows for large, uncompressed media files to be played in real time, and thus brings hi-fidelity reproduction to streaming media.

[0116] The client side software technology may choose to preferentially download from those nodes offering the highest current throughput for a particular client at his location, resulting in further speed improvements to throughput. From the entire worldwide pool of available nodes, each client application may choose to read from media streams from those nodes which offer the highest throughput at the moment. The redundancy of erasure coding also means that more than one node contains the next needed fragments, allowing the client to choose the highest throughput nodes available.

[0117] The dispersal of data fragments to data storage nodes can also be optimized based on the current throughput conditions. Nodes with the best connectivity can be chosen to store larger amounts of data fragments, thus optimizing the storage nodes available for maximum speed of data transfer during the dispersal process.

[0118] Specifically, the erasure coding used in the technology may be done at the server side, on servers that have been chosen for high performance, since erasure coding can be a CPU intensive task.

[0119] B. Data security

[0120] As discussed above, the distributed and "virtual erasure coding" streaming technique disclosed herein offers vast improvements of data security over prior streaming technology which stores a whole file in a single physical cloud storage location.

[0121] Further, the servers used for both processing and storage of file slice fragments may be shared by multiple clients, with no way for a hacker to identify from the slices to which client it belongs. This makes it even more difficult for a hacker to compromise the security of media file data stored using this technology.

[0122] C. Data availability

[0123] As discussed above, the distributed storage and "virtual erasure coding" streaming technique disclosed herein also offers improvement in the availability of the data, compared to prior art streaming technology. By splitting the file into multiple file slice fragments which are stored on a number of physical nodes, that preferably are located at different locations, communication problems between the client location and one of the physical nodes may be offset by normal communications with the other data locations. The overall effect of having multiple locations is to insulate the system from outages due to communications disruptions at one of the sites.

[0124] The use of erasure coding that makes the original data unrecognizable, and multiple nodes with redundant data adds powerful and secure error correcting technology. Packet loss problems, which plague the prior art streaming technology are no longer a relevant consideration. The prior art streaming technology must often put multiple copies of the same media file on many servers throughout the geographical service area, to make sure that each client has good connectivity to the server that stores the data stream he wishes to play. The disclosed streaming technology eliminates the need for full redundant copies of the original media file on multiple servers throughout the service area.

[0125] D. Data reliability

[0126] The distributed storage and "virtual erasure coding" streaming technology disclosed herein also brings vast improvements in reliability of streaming media over the prior art. Separation of each file into file slice fragments means that hardware or software failures or errors at one of the physical server storage locations will not eliminate access to the file, as is the case when the entire file is stored in one physical location, as in the prior art technology. Erasure coding technology for making the original data unrecognizable insures high quality error correction capabilities while enhancing security of the media content.

[0127] E. Digital Rights Management Security

[0128] The protection of digital rights (DRM) is a particularly important issue with streaming media files. Many third-party products are available which can circumvent DRM protection schemes in streaming media. As the disclosed technology breaks up the data stream into data fragments which may be encrypted and each processed with erasure coding that can make the original data unrecognizable, DRM protection schemes are greatly enhanced. If the client requesting the streaming media does not have rights to the file itself, but only rights to play the file, the encrypted and erasure-coded data fragments do not have to be physically assembled into an actual media file on the client device, even during play. This invites much stronger DRM schemes which cannot be readily circumvented by the usual third party technologies in use today.

[0129] To summarize, in an exemplary embodiment, the distributed storage and "virtual erasure coding" streaming technology disclosed herein accomplishes the following fundamental tasks:

1) Splitting of a content provider's media file slice into pieces or file slices which will eventually be broken up further into file fragments that are erasure coded on distributed erasure coding servers to provide unrecognizable pieces.

2) Creation of maps of the file slices which describe how the files were split to allow for re-assembly of the data at the client. This map is stored in a metadata file.

3) Optional encryption of the file slices for additional data security.

4) Optional compression of the file slices to reduce the size of data storage and improve transfer speed.

5) Erasure coding of the file slices to enable enhanced error correction and data recovery. The slices are divided into file slice fragments by the erasure coding process.

6) Creation of a map of the file slice fragments needed to reassemble them into file slices. This map is stored in a second metadata file.

7) Optional encryption of the file slice fragments for additional data security.

8) Optional compression of the file slice fragments to reduce storage space requirements and improve transfer speed.

9) Decoding on the client device of the file slice fragments and re-assembly into file slices, and then into the whole media file, for playing on the client media player (or browser). Note that the fragments must be assembled into slices in the proper order, and the slices must be assembled into the whole file in the proper order. The client software uses the mapping information provided by the two metadata files to reassemble the media file in these two stages.

[0130] The basic structure of this technology may be visualized as being implemented by the following four layers:

[0131] 1. The CSP (see, FIG. 1) slices the content provider's media file into file slices, optionally encrypts the slices, and generates a meta-data file with a map of how the slices can be re-assembled into the original media file. The meta-data file also maintains information on the order of each file slice needed to assemble the slices in the proper order.

[0132] 2. The FEDP (see, FIG. 1) breaks each file slice into file slice fragments using erasure coding that produces unrecognizable pieces. In an exemplary embodiment erasure coding adds 30% of data redundancy. A second meta-data file maps how the file slice fragments are reassembled into to file slices. The second meta-data file also maintains information on the order of each fragment needed to assemble the slices in the proper order, during playing of the fragments on the client device.

[0133] 3. The SNNs (see, FIG. 1) are the various storage nodes used to disperse the data fragments. The storage nodes are not necessarily all servers in the cloud. The nodes may be a data center, a hard disk in a computer, a mobile device, or some other

multimedia device capable of data storage. The number and identity of these storage nodes can be selected by the content provider to optimize the latency and security of the storage configuration with nodes having the lowest average latency and best availability.

[0134] 4. An end-user client decoder (ECD) that may be implemented on top of current technology streaming media player software. This fourth layer initiates a request to the content provider for streaming media, and then receives mapping files derived from the two meta-data files formed in layers (1) and (2), above which allow the ECD to assemble the file slice fragments into slices, and the slices into the original media file, for the playback or storage of the media file. As evident, the media file must be assembled in the proper order needed for on demand playing of the media content. If the client has purchased rights to the streamed media for downloading the complete file, the ECD will both play and assemble the original media file, once it has completely downloaded. If the client only has rights to play the media file, the ECD will only play the media file in the proper order, while storing the file slice fragments for possible re-play, without ever assembling them into a complete file. The ECD will also buffer the data fragments in storage on the client device if the rate of download exceeds the rate of media play, which should happen most of the time. The ECD may also interact with the media player to receive and process requests for media file segments which are located ahead of or behind the current time of media file play.

[0135] Additional Performance Considerations

[0136] If the particular media file is in high demand from multiple clients, there are two main approaches that can be taken to meet the increased demand:

[0137] First, a larger number of fragment storage nodes may be employed for dispersal of the erasure encoded data fragments. If the demand is primarily coming from one geographic area, nodes could be chosen for dispersal with the best data throughput rates for clients in that area.

[0138] Second, a higher level of redundancy may be chosen for the erasure coding step. For example, instead of 30% redundancy, higher levels of redundancy will help ensure greater available under load.

[0139] These two steps may be performed dynamically to meet specific demand and load requirements as they occur in real time.

[0140] In addition, certain slices or fragments may be singled out for greater levels of redundancy to improve availability. Specifically, the first segments of the media file could should be given the highest level of redundancy to meet the needs of increased demand.

[0141] Although the disclosed subject matter has been described and illustrated with respect to certain exemplary embodiments thereof, it should be understood by those skilled in the art that features of the disclosed embodiments can be combined, rearranged, and modified, to produce additional embodiments within the scope of the disclosure, and that various other changes, omissions, and additions may be made therein and thereto, without departing from the spirit and scope of the present invention.

WHAT IS CLAIMED IS:

1. A method of processing media content, comprising the steps of:

separating the media content into a plurality of file slices;

generating metadata for the reassembly of media content from the file slices;

erasure coding the file slices, wherein the slices are divided into discrete file slice fragments;

generating metadata for the reassembly of the file slices from the file slice fragments; and

sending the file slice fragments to a plurality of dispersed networked storage nodes, wherefrom the media content may be retrieved and reconstructed using the metadata.

2. The method of claim 1 wherein the media content is not recognizable from the erasure-coded file slice fragments.

3. The method of claim 2 wherein the step of erasure coding is performed across a plurality of data processors.

4. The method of claim 2, further comprising the steps of:

receiving at a client decoder the file slice fragments from the networked storage nodes; and

reconstructing the media content according to the metadata.

5. The method of claim 4 wherein the media content is one of streaming video and audio content, and wherein the step of reconstructing the media content is performed contemporaneously during playback of the media content.

6. The method of claim 5 wherein the steps of receiving and reconstructing are performed in response to a client request for the media content; and/or wherein each file slice fragment is assigned a unique identifier and the metadata indicates the location of each file slice fragment in the plurality of dispersed networked storage nodes based on its unique identifier; and/or wherein the step of erasure coding results in at least a thirty percent data redundancy level.

7. The method of claim 6, third alternative, wherein the number and identify of the storage nodes are selected by a content provider to reduce the latency of the storage node network.

8. The method of claim 1 wherein the storage nodes are located in physically separated devices.

9. The method of claim 8 wherein the physically separated devices are geographically dispersed.

10. The method of claim 1 wherein no one storage node has sufficient information to allow reconstruction of the media content.

11. A method of receiving media content, comprising the steps of:

requesting media content stored across a plurality of dispersed networked storage nodes as erasure-coded file slice fragments;

receiving at a client decoder the erasure-coded file slice fragments and metadata containing information for reconstruction of the media content from the file slice fragments; and

reconstructing the media content at the client decoder from the file slice fragments based on the metadata.

12. The method of claim 11 wherein the media content is one of streaming video and audio content.

13. The method of claim 12 wherein the media content is unrecognizable from the file slice fragments.

14. The method of claim 11 wherein each file slice fragment is assigned a unique identifier that indicates the location of the file slice fragment in the plurality of dispersed networked storage nodes; and/or wherein the number and identify of the storage nodes are selected by a content provider to reduce the latency of the storage node network.

15. The method of any of the proceeding claims, wherein the file slices are encrypted prior to erasure coding; and/or

wherein file slices are compressed prior to the step of erasure coding in the method of processing media content.

16. A method for distributed processing and storage of data, comprising the steps of:

dividing a data file into a plurality of file slices;

providing a plurality of data processors for receiving the file slices, each data processor erasure coding at least one of the file slices to generate a plurality of unrecognizable file slice fragments;

storing the file slice fragments in a network of storage nodes, wherein no one storage node has sufficient information to allow reconstruction of the data file.

17. The method of claim 16 wherein the step of erasure coding divides a file slice having m segments into a plurality of n unrecognizable file slice fragments, where n>m, by using a data mixer algorithm that permits reconstruction of the n file slice fragments from any m file slice fragments.

18. The method of claim 17 wherein the data mixer algorithm uses a Cauchy matrix as a generator matrix; or wherein the data mixer algorithm uses a Vandermonde matrix as a generator matrix.

19. The method of claim 5 wherein the steps of receiving and reconstructing are performed in response to a client request for the media content.

20. The method of claim 5 wherein each file slice fragment is assigned a unique identifier and the metadata indicates the location of each file slice fragment in the plurality of dispersed networked storage nodes based on its unique identifier.

21. The method of claim 5 wherein the step of erasure coding results in at least a thirty percent data redundancy level.

22. The method of claim 21 wherein the number and identify of the storage nodes are selected by a content provider to reduce the latency of the storage node network.

23. The method of claim 1 wherein the file slices are encrypted prior to the step of erasure coding.

24. The method of claim 1 wherein file slices are compressed prior to the step of erasure coding.

25. The method of claim 11 wherein each file slice fragment is assigned a unique identifier that indicates the location of the file slice fragment in the plurality of dispersed networked storage nodes.

26. The method of claim 11 wherein the number and identify of the storage nodes are selected by a content provider to reduce the latency of the storage node network.

27. The method of claim 11, wherein the file slices are encrypted prior to erasure coding.

28. The method of claim 17 wherein the data mixer algorithm uses a Cauchy matrix as a generator matrix.

29. The method of claim 17 wherein the data mixer algorithm uses a Vandermonde matrix as a generator matrix.

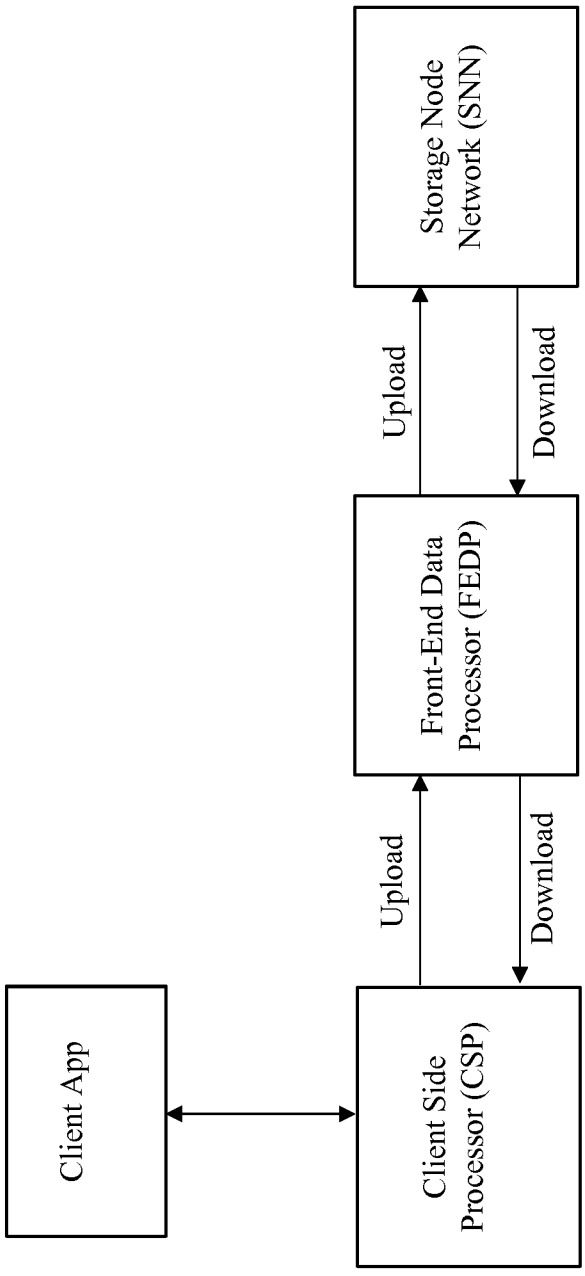

Storage Node Network (SNN)
Upload
Download
Front-End Data Processor (FEDP)
Upload
Download
Client App
Client Side Processor (CSP)

FIG. 1

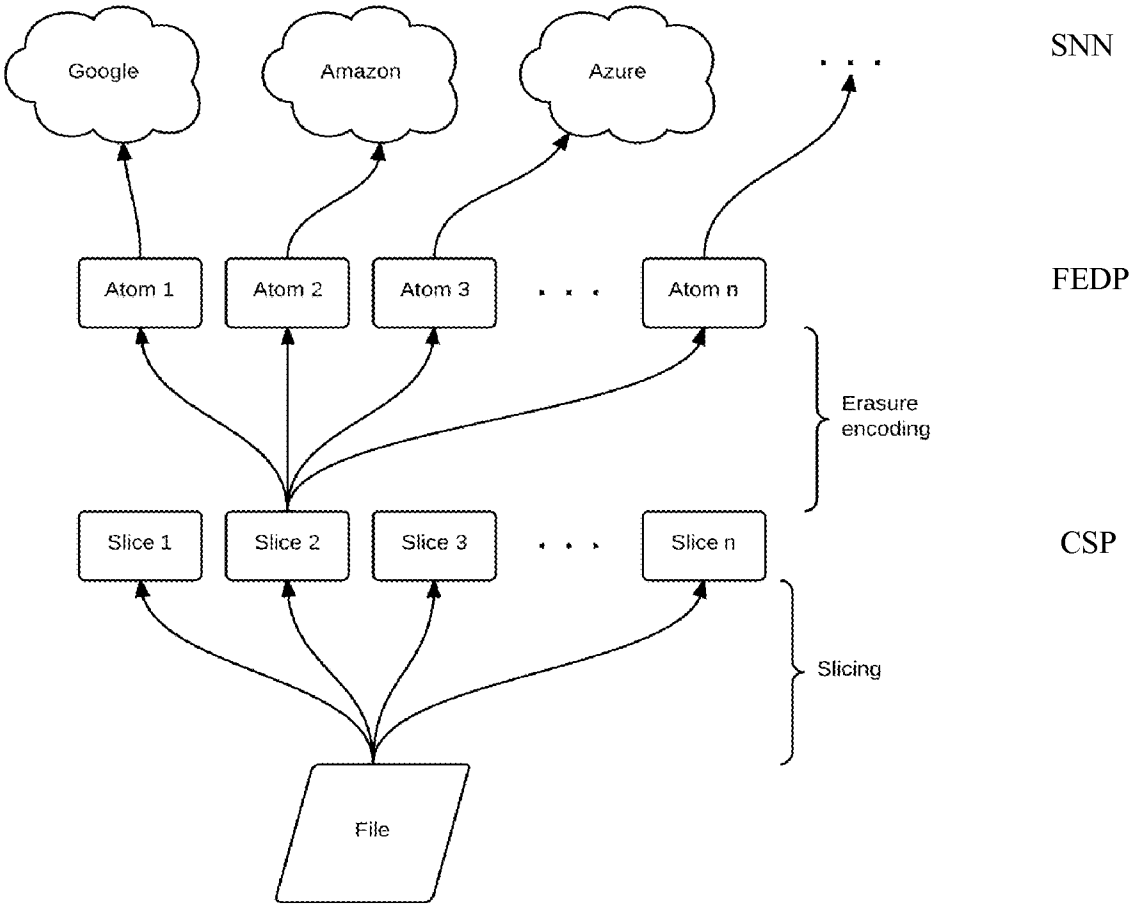

Google
Amazon
Azure
. . .
SNN
Atom 1
Atom 2
Atom 3
. . .
Atom n
FEDP
Erasure encoding
Slice 1
Slice 2
Slice 3
. . .
Slice n
CSP
Slicing
File

FIG. 2

1	Client app sends requests to upload file to CSP for file in client datacenter. The client app also sends to the CSP the client's list of optimized SNN's for use later in storing the file fragments. A metadata file *metadata0* is created to describe the files physical parameters in the client's datacenter (size, location, checksum, etc.). This metadata file is encrypted and stored in the client's datacenter and also sent to storage nodes.	Whole file in data center	Client app
2	CSP assigns unique file ID (FID) to the file about to be upload	Whole file in data center	CSP
3	CSP slices file into n slices each of size x. The number of slices n and the size x, based on the size of the whole file is a parameter which can be configured by the client app, and is passed to the CSP. A typical value for the slicing: a 1Gb file is sliced into 500 slices. Presently, the slices are of equal size, although a future enhancement will see more intelligent slicing based on a more complicated algorithm to optimize client security. Each slice contains its raw data and the information on client parameters on which SNN's to use for storing this file, as transmitted previously by the client app.	*n* slices of size *X*	CSP
4	Each slice is assigned a unique slice ID (SID) which is used to map the slices for reassembly	*n* slices of size *x*	CSP
5	A second metadata file *metadata1* is created which maps how these slices with their unique slice ID (SID) are reassembled to the original file. This metadata file is encrypted using the client's encryption key, and stored at the client's datacenter, it is sent into storage nodes as well.	*metadata1,encrypted,* maps slices	CSP
6	Each slice is encrypted using the client's encryption key. Only the client can read the slice. Symmetric encryption is used.	*n* slices of size *x* encrypted	CSP
7	The CSP sends the encrypted slices on the FEDP's. Latency optimization is performed at the CSP in choosing which FEDP's to use for processing the slices. This is necessary as the client's data centers may lack optimal speed and availability in regards to the FEDP's. Once chosen, the slices are sent out randomly to FEDP's for the next stage of processing.	*n* slices of size *x* encrypted	CSP
8	FEDP's are data processing servers maintained by the inventions' servers for the next stage of processing, with highly optimized performance, latency and availability to the SNN's. Optimally, there should be twice as many FEDP's as SNN's in a given client setup to optimize the speed and security improvements in the invention's technology. Normally, these FEDP's will be located in the cloud, outside the client's data center, although the client could also specify that FEDP's would also be at the client's own site to optimize performance. The number of FEDP servers used is optimized for the size of the files transferred, and the available bandwidth.	*n* slices of size *x* encrypted, normally outside client's site.	FEDP

FIG. 3

9	Each FEDP uses Reed-Solomon Erasure coding to split up each slice into slice fragments adding some data used in the Erasure coding error detection and correction schemes. A typical example from above, our 1 Gb. file was split initially (at the CSP) into 500 equal size slices, and each slice is split at an FEDP into 15 slice fragments using Erasure coding. Erasure coding is a very CPU intensive process, and these servers have been built with optimum performance for this processing.	*a* slice fragments per slice created for each slice using Erasure coding.	FEDP
10	Each slice fragment is given an "Fragment ID", and assigned to one of the previously designated SNN data centers. A metadata file *metadata2* is created which maps each of the slice fragments to the slice from which it was generated, and on which SNN it is to be stored. This metadata file is encrypted using the server's encryption key, and sent to each of the SNN's used in storing the slice fragments which make up the slices, which make up the original file. Thus, each SNN contains a redundant copy of *metadata2*.	Metadata file *metadata2* is created to map all the slice fragments to their original slices, and sent to each SNN.	FEDP
11	Each of the slice fragments is encrypted using the server's encryption key, and sent to the SNN designated for those slice fragments. There is no need for latency optimization between the FEDP's and SNN's, as FEDP's are high performance, high availability servers, pre-optimized for latency with the set of chosen SNN's.	*a* slice fragments per slice	FEDP->SNN
12	Each SNN stores a complete *metadata2* map file needed to reassemble the slices, and the slice fragments assigned to that SNN. For speed, we do not need a very large number of SNN, with 5 SNN's optimizing the speed performance enhancements, but we want a much larger number of SNN's for security improvements, as slice fragments are spread throughout a larger number of SNN's. This could be at least 20 SNN's, or even 50 SNN's for extra security.	*a* slice fragments per slice	SNN
13	SNN's can be carefully rated as to their latency performance, relative to the invention's system to aid the user in defining which SNN's should be chosen for file dispersal.		SNN

FIG. 3 (CONT.)

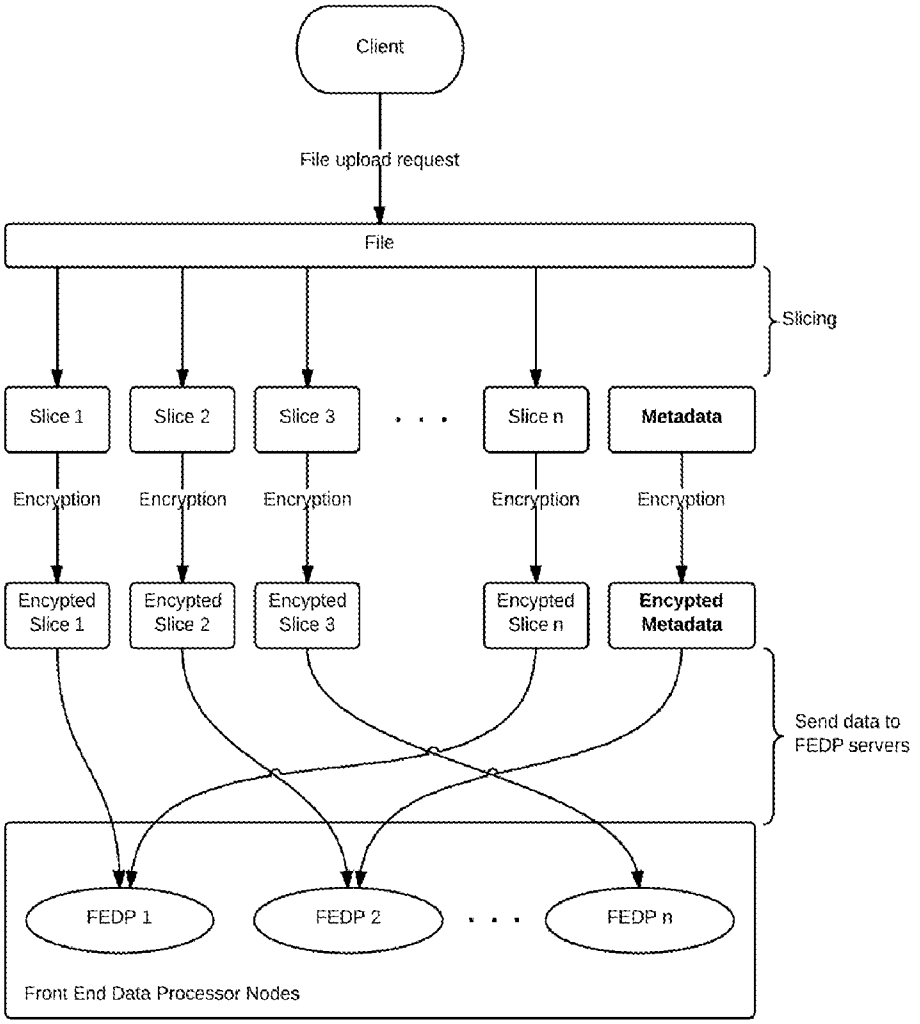

Client
File upload request
File
Slicing
Slice 1
Slice 2
Slice 3
Slice n
Metadata
Encryption
Encryption
Encryption
Encryption
Encryption
Encypted Slice 1
Encypted Slice 2
Encypted Slice 3
Encypted Slice n
Encypted Metadata
Send data to FEDP servers
FEDP 1
FEDP 2
FEDP n
Front End Data Processor Nodes
Client Front End
CSP
FEDP

FIG. 4A

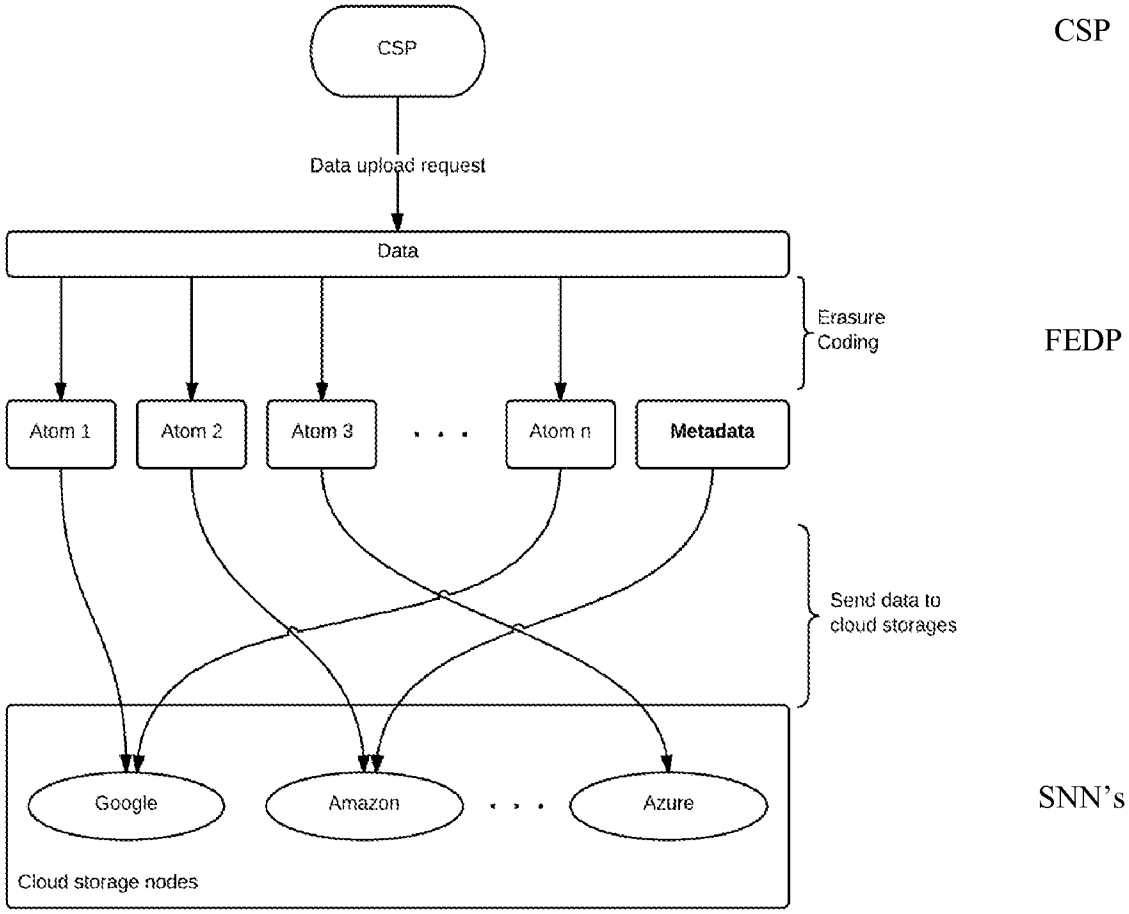

CSP
Data upload request
Data
Erasure Coding
FEDP
Atom 1
Atom 2
Atom 3
Atom n
Metadata
Send data to cloud storages
Google
Amazon
Azure
SNN's
Cloud storage nodes

FIG. 4B

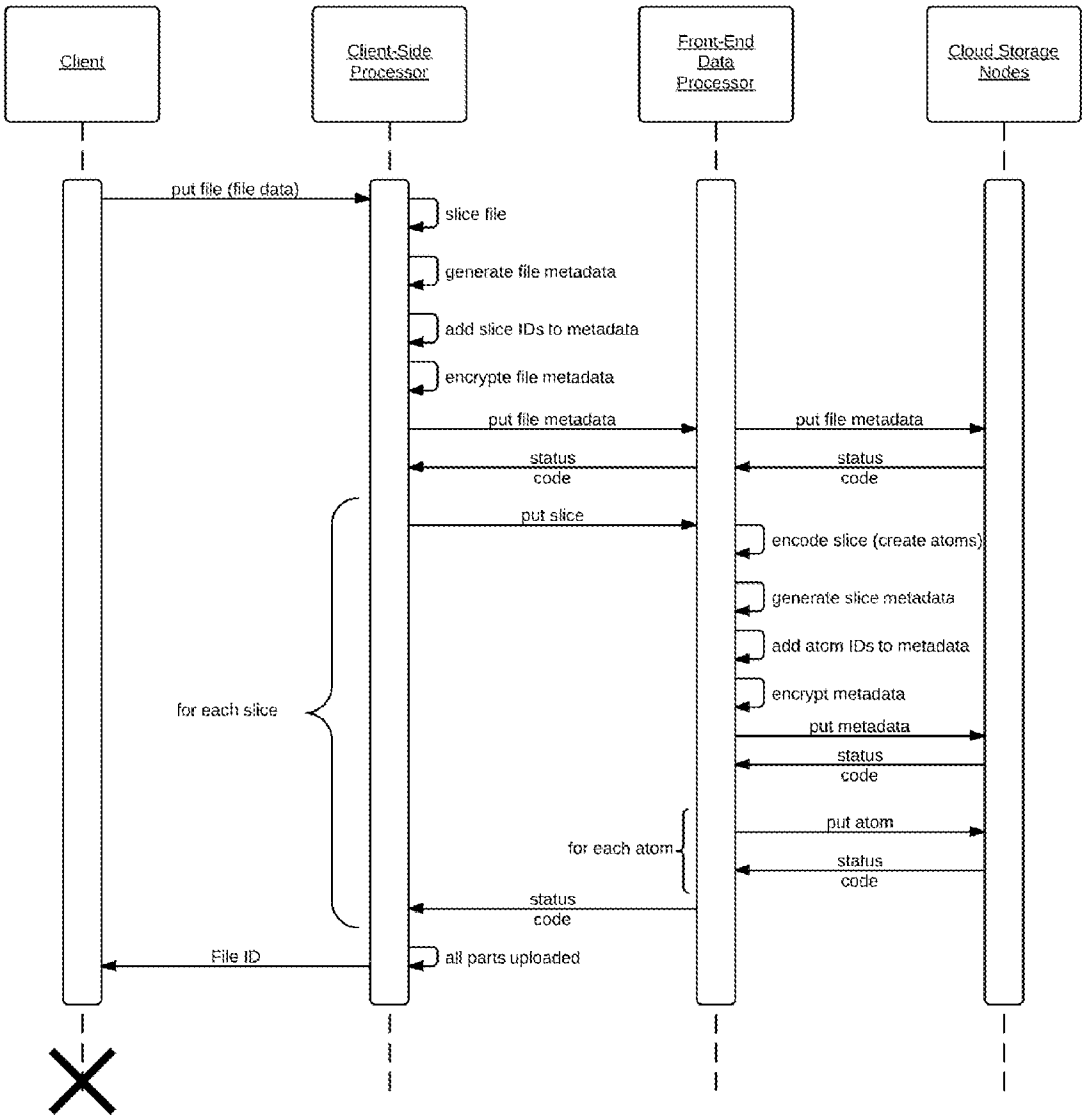

Client
Client-Side Processor
Front-End Data Processor
Cloud Storage Nodes
put file (file data)
slice file
generate file metadata
add slice IDs to metadata
encrypte file metadata
put file metadata
put file metadata
status code
status code
put slice
encode slice (create atoms)
generate slice metadata
add atom IDs to metadata
encrypt metadata
for each slice
put metadata
status code
put atom
for each atom
status code
status code
File ID
all parts uploaded

FIG. 5

1	Client app sends requests to download file to CSP for file that was previously uploaded. The client app sends the file id FID of the file to be downloaded to the CSP.	Uploaded file stored in SNN's	Client app
2	The CSP also receives *metadata1* which contains the map of slices to be reassembled into the whole file. This metadata file must be unencrypted using the client encryption key.	*metadata1* from client	CSP
3	The CSP needs to receive its file slices from the FEDP. The CSP sends the FEDP a request for file slice slice ID SID *x* .	Request slice SID *x*	CSP
4	THE FEDP receives a request for file slice SID *x* which causes it to send a request to one of the SNN's to receive metadata file *metadata2*. This metadata file must be unencrypted using the server's encryption key.	FEDP receives *metadata2*	FEDP
5	Once the FEDP has received and unencrypted *metadata2*, it has a map of where to find the slice fragments corresponding to this slice id SID *x* on the various SNN data centers.	FEDP requests slice fragments from slice SID	FEDP
6	The FEDP now sends out requests to all the SNN's which contain the slice fragments of file slice SID *x* , to retrieve these slice fragments according to their Fragment ID's. The FEDP makes use of the Erasure coding built into all the slice fragments to retrieve these fragments with error checking and error correction to insure that the slice fragments have been faithfully received by the FEDP.	SNN's transmit slice fragments to FEDP	FEDP
7	The FEDP decrypts all the slice fragments using the server's encryption key, and reassembles them into file slice SID *x*.	FEDP reassembles slice SID	FEDP
8	The FEDP transmits file slice SID *x* back to the CSP, as per the CSP's request, assuming all the error checks have passed.	FEDP transmits slice back to CSP	FEDP
9	The CSP takes the file slice SID *x* which it just received from the FEDP, and unencrypts it using the client's encryption key.	CSP unencrypts slice SID	CSP
10	The CSP uses the metadata file *metadata1* to request the next file slice SID *y* from the FEDP's.	CSP request next slice from FEDP	CSP
11	The FEDP repeats steps *4-8* above to assemble file slice *y* from the slice fragments stored on the SNN's until it has reassembled the requested file slice SID *y*, and sends this back to the CSP	CSP continues to request all the slices in *metadata1*	FEDP
12	The CSP unencrypts file slice SID *y* as above in step *9*. The steps are repeated for each file slice SID contained *in metadata1*, with the CSP requesting assembled file slices from the FEDP, which assembles them from slice fragments dispersed among the SNN's.	CSP and FEDP repeat steps until all slices are retrieved by CSP.	CSP,FEDP

FIG. 6

13	When the CSP has received all the slices mapped in *metafile1*, it uses *metafile1* to reassemble the original, complete uploaded file, FID.	CSP reassembles all the slices into the original file FID, using *metadata1*	CSP
14	The CSP send the complete, reassembled, unencrypted file back to the client's data center, completing the client app's download request.	CSP transmits reassembled file to client data center	CSP

FIG. 6 (CONT.)

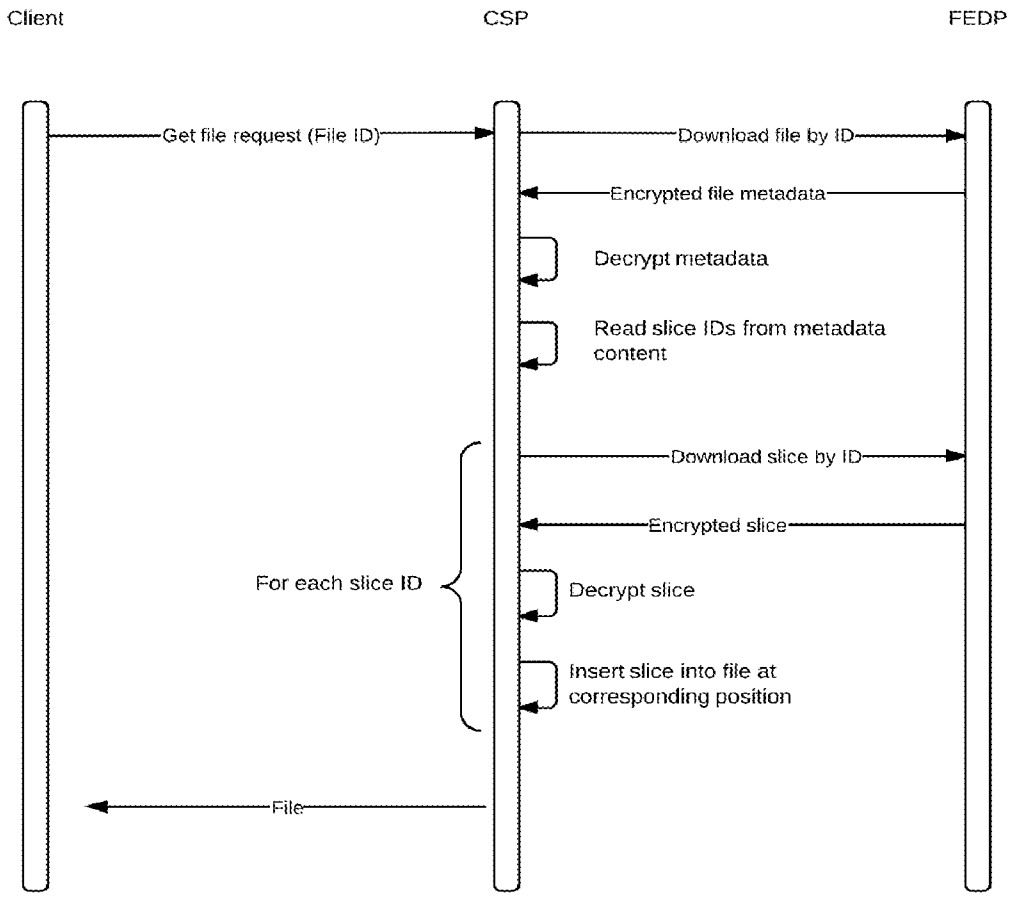

Client
CSP
FEDP
Get file request (File ID)
Download file by ID
Encrypted file metadata
Decrypt metadata
Read slice IDs from metadata content
Download slice by ID
Encrypted slice
For each slice ID
Decrypt slice
Insert slice into file at corresponding position
File

FIG. 7A

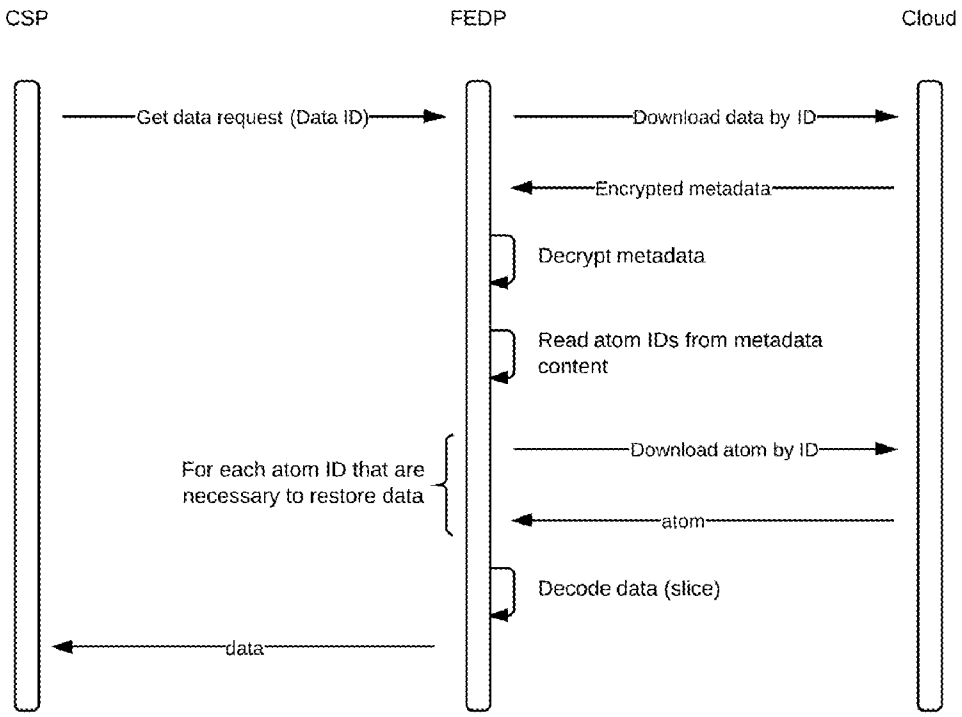

CSP
FEDP
Cloud
Get data request (Data ID)
Download data by ID
Encrypted metadata
Decrypt metadata
Read atom IDs from metadata content
For each atom ID that are necessary to restore data
Download atom by ID
atom
Decode data (slice)
data

FIG. 7B

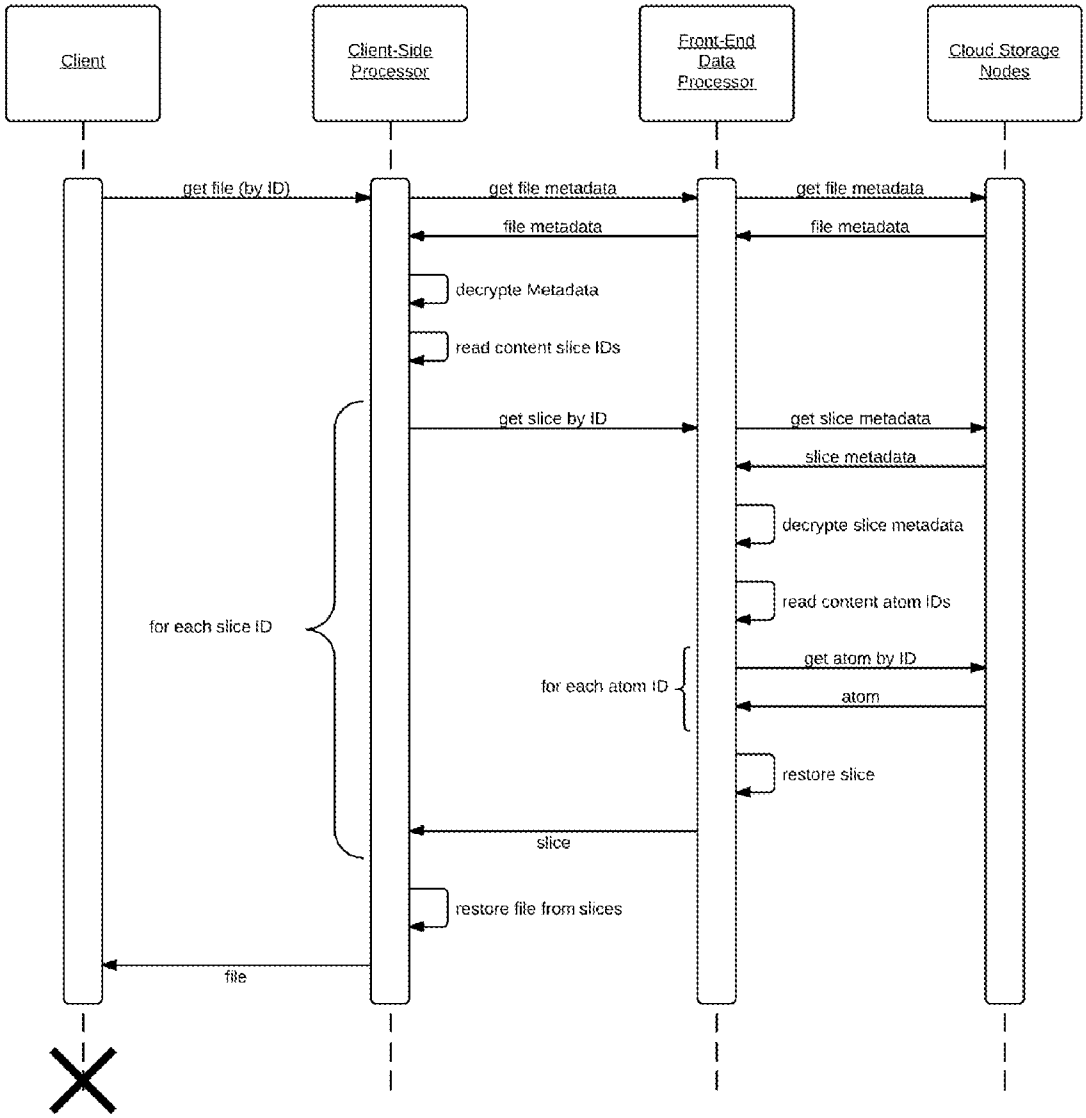

Client
Client-Side Processor
Front-End Data Processor
Cloud Storage Nodes
get file (by ID)
get file metadata
get file metadata
file metadata
file metadata
decrypte Metadata
read content slice IDs
get slice by ID
get slice metadata
slice metadata
decrypte slice metadata
read content atom IDs
for each slice ID
get atom by ID
for each atom ID
atom
restore slice
slice
restore file from slices
file

FIG. 8

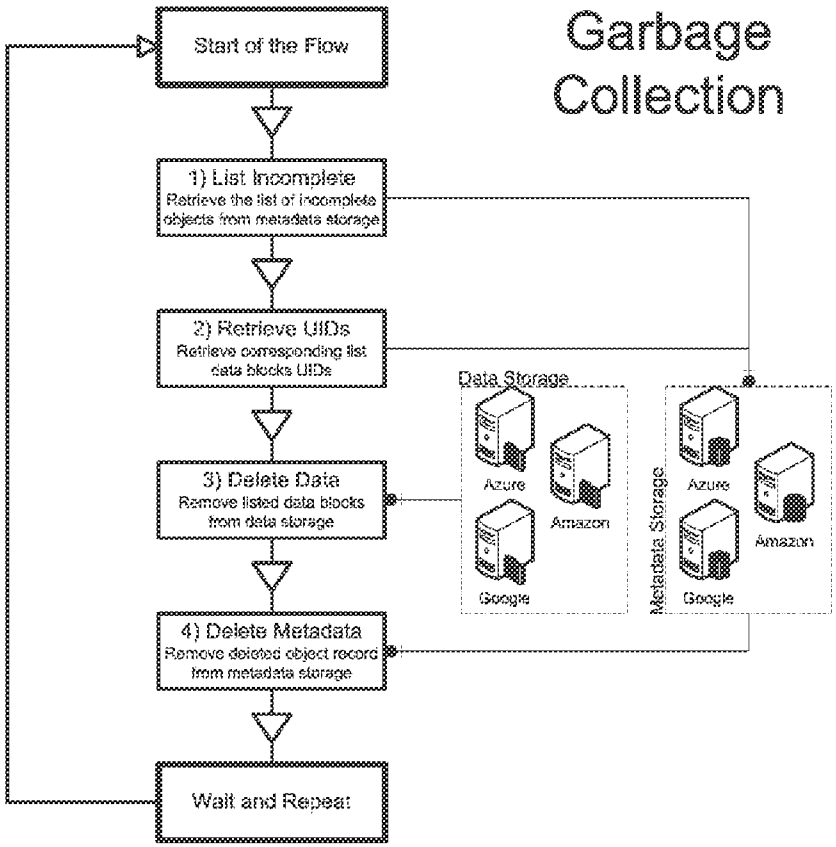

Garbage Collection
Start of the Flow
1) List Incomplete
Retrieve the list of incomplete objects from metadata storage
2) Retrieve UIDs
Retrieve corresponding list data blocks UIDs
3) Delete Data
Remove listed data blocks from data storage
4) Delete Metadata
Remove deleted object record from metadata storage
Wait and Repeat
Data Storage
Azure
Amazon
Google
Metadata Storage
Azure
Amazon
Google

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/030163

A. CLASSIFICATION OF SUBJECT MATTER
IPC (2015.01) G06F 17/30, H03M 13/00

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 (2015.01) G06F 17/30, H03M 13/00

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)
Databases consulted: PATENTSCOPE, THOMSON INNOVATION, Google Patents, FamPat database
Search terms used: dispersed, distributed, storage, nodes, erasure coding, FEC, files, media, sliced, fragmented, contemporaneously, simultaneously, reconstruct,retrieve, assemble, media, video, audio, streaming,metadata, identifier, latency, content, sevice, provider

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2011161666 A1 GLADWIN et al. 30 Jun 2011 (2011/06/30) abstract, ¶ 0047,0052,0053, 0064, 0070, 0090, 0107	11,12,14,25,27
Y	abstract ¶ 0035, 0046, 0047 0052-0054, 0064, 0068-0070, 0082, 0089 , 0090, 0107	1-4,8-10,13,15-18, 23,24,26,28,29
Y	US 2012060072 A1 SIMITCI et al. 08 Mar 2012 (2012/03/08) ¶ 0019, 0037, 0038, 0058	1-4,8-10,13,15-18, 23,24,26,28,29
Y	US 2010218037 A1 SWARTZ et al. 26 Aug 2010 (2010/08/26) ¶ 0013	18,28,29
A	US 2007079083 A1 GLADWIN et al. 05 Apr 2007 (2007/04/05) Whole document	1-29

[X] Further documents are listed in the continuation of Box C. [X] 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	Date of mailing of the international search report
01 Sep 2015	01 Sep 2015
Name and mailing address of the ISA: Israel Patent Office Technology Park, Bldg.5, Malcha, Jerusalem, 9695101, Israel Facsimile No. 972-2-5651616	Authorized officer COPPENHAGEN Uri Telephone No. 972-2-5657811

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/030163

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2011219100 A1 DHUSE et al. 08 Sep 2011 (2011/09/08) Whole document	1-29
A	WO 2014005279 A1 NIV PEKING SHENZHEN GRAD SCHO?[CN]; 09 Jan 2014 (2014/01/09) Whole document	1-29
A	Rizzo, L. (1997). Effective erasure codes for reliable computer communication protocols. ACM SIGCOMM computer communication review, 27(2), 24-36.Retrieved from the internet on 31-08-2015 <http://ccr.sigcomm.org/archive/1997/apr97/ccr-9704-rizzo.pdf> Rizzo, L. 31 Dec 1997 (1997/12/31) Whole document	1-29

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
US 2011161666 A1	30 Jun 2011	US 2011161666 A1	30 Jun 2011
		US 2011161781 A1	30 Jun 2011
		US 8352831 B2	08 Jan 2013
		US 2013125190 A1	16 May 2013
		US 8612827 B2	17 Dec 2013
		US 2014317694 A1	23 Oct 2014
		US 2014328573 A1	06 Nov 2014
		US 2014331103 A1	06 Nov 2014
		US 2014331330 A1	06 Nov 2014
US 2012060072 A1	08 Mar 2012	US 2012060072 A1	08 Mar 2012
		US 8473778 B2	25 Jun 2013
US 2010218037 A1	26 Aug 2010	US 2010218037 A1	26 Aug 2010
		EP 2342661 A1	13 Jul 2011
		EP 2342661 A4	20 Feb 2013
		US 2012185437 A1	19 Jul 2012
		US 9098519 B2	04 Aug 2015
		WO 2010033644 A1	25 Mar 2010
		WO 2012068184 A1	24 May 2012
US 2007079083 A1	05 Apr 2007	US 2007079083 A1	05 Apr 2007
		US 7574579 B2	11 Aug 2009
		EP 2008185 A2	31 Dec 2008
		EP 2008185 A4	05 May 2010
		EP 2201460 A1	30 Jun 2010
		EP 2201460 A4	22 Aug 2012
		EP 2201461 A1	30 Jun 2010
		EP 2201461 A4	22 Aug 2012
		EP 2201469 A1	30 Jun 2010
		EP 2201469 A4	15 Jul 2015

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		EP 2260387 A2	15 Dec 2010
		EP 2460080 A2	06 Jun 2012
		JP 2009533759 A	17 Sep 2009
		US 2007079082 A1	05 Apr 2007
		US 7546427 B2	09 Jun 2009
		US 2007174192 A1	26 Jul 2007
		US 7574570 B2	11 Aug 2009
		US 2009254720 A1	08 Oct 2009
		US 7818518 B2	19 Oct 2010
		US 2009094251 A1	09 Apr 2009
		US 7904475 B2	08 Mar 2011
		US 2010179966 A1	15 Jul 2010
		US 7953771 B2	31 May 2011
		US 2007079081 A1	05 Apr 2007
		US 7953937 B2	31 May 2011
		US 2010017531 A1	21 Jan 2010
		US 2011125916 A9	26 May 2011
		US 7962641 B1	14 Jun 2011
		US 2010023529 A1	28 Jan 2010
		US 8019960 B2	13 Sep 2011
		US 2010063911 A1	11 Mar 2010
		US 8140777 B2	20 Mar 2012
		US 2009094318 A1	09 Apr 2009
		US 8171101 B2	01 May 2012
		US 2010115063 A1	06 May 2010
		US 8171102 B2	01 May 2012
		US 2010169415 A1	01 Jul 2010
		US 8185614 B2	22 May 2012
		US 2011202568 A1	18 Aug 2011
		US 8190662 B2	29 May 2012
		US 2010250751 A1	30 Sep 2010

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 8200788 B2	12 Jun 2012
		US 2009094320 A1	09 Apr 2009
		US 8209363 B2	26 Jun 2012
		US 2010023710 A1	28 Jan 2010
		US 8244991 B2	14 Aug 2012
		US 2010023524 A1	28 Jan 2010
		US 8285878 B2	09 Oct 2012
		US 2010306578 A1	02 Dec 2010
		US 8352782 B2	08 Jan 2013
		US 2011077086 A1	31 Mar 2011
		US 8357048 B2	22 Jan 2013
		US 2011289383 A1	24 Nov 2011
		US 8448044 B2	21 May 2013
		US 2012290873 A1	15 Nov 2012
		US 8464096 B2	11 Jun 2013
		US 2012246421 A1	27 Sep 2012
		US 8468311 B2	18 Jun 2013
		US 2010169500 A1	01 Jul 2010
		US 8478865 B2	02 Jul 2013
		US 2011107165 A1	05 May 2011
		US 8479078 B2	02 Jul 2013
		US 2011029836 A1	03 Feb 2011
		US 8489915 B2	16 Jul 2013
		US 2011107094 A1	05 May 2011
		US 8522022 B2	27 Aug 2013
		US 2010169391 A1	01 Jul 2010
		US 8533256 B2	10 Sep 2013
		US 2011072210 A1	24 Mar 2011
		US 8549351 B2	01 Oct 2013
		US 2011029809 A1	03 Feb 2011
		US 8555109 B2	08 Oct 2013

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 2010161916 A1	24 Jun 2010
		US 8560882 B2	15 Oct 2013
		US 2011072321 A1	24 Mar 2011
		US 8572429 B2	29 Oct 2013
		US 2011107026 A1	05 May 2011
		US 8589637 B2	19 Nov 2013
		US 2011029731 A1	03 Feb 2011
		US 8595435 B2	26 Nov 2013
		US 2011289565 A1	24 Nov 2011
		US 8621580 B2	31 Dec 2013
		US 2012030736 A1	02 Feb 2012
		US 8627114 B2	07 Jan 2014
		US 2013254624 A1	26 Sep 2013
		US 8627177 B2	07 Jan 2014
		US 2010287200 A1	11 Nov 2010
		US 8630987 B2	14 Jan 2014
		US 2013290809 A1	31 Oct 2013
		US 8631303 B2	14 Jan 2014
		US 2012198203 A1	02 Aug 2012
		US 8688949 B2	01 Apr 2014
		US 2011219100 A1	08 Sep 2011
		US 8694668 B2	08 Apr 2014
		US 2012198197 A1	02 Aug 2012
		US 8694752 B2	08 Apr 2014
		US 2011029711 A1	03 Feb 2011
		US 8706980 B2	22 Apr 2014
		US 2013275833 A1	17 Oct 2013
		US 8713358 B2	29 Apr 2014
		US 2011106855 A1	05 May 2011
		US 8732206 B2	20 May 2014
		US 2011106904 A1	05 May 2011

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 8769035 B2	01 Jul 2014
		US 2011016122 A1	20 Jan 2011
		US 8819011 B2	26 Aug 2014
		US 2011071988 A1	24 Mar 2011
		US 8819179 B2	26 Aug 2014
		US 2014040662 A1	06 Feb 2014
		US 8819482 B2	26 Aug 2014
		US 2013275834 A1	17 Oct 2013
		US 8819516 B2	26 Aug 2014
		US 2014053018 A1	20 Feb 2014
		US 8826067 B2	02 Sep 2014
		US 2012027134 A1	02 Feb 2012
		US 8842746 B2	23 Sep 2014
		US 2011161681 A1	30 Jun 2011
		US 8856552 B2	07 Oct 2014
		US 2011286594 A1	24 Nov 2011
		US 8861727 B2	14 Oct 2014
		US 2013275844 A1	17 Oct 2013
		US 8868969 B2	21 Oct 2014
		US 2011289366 A1	24 Nov 2011
		US 8874868 B2	28 Oct 2014
		US 2008183975 A1	31 Jul 2008
		US 8880799 B2	04 Nov 2014
		US 2014032961 A1	30 Jan 2014
		US 8880940 B2	04 Nov 2014
		US 2013102395 A1	25 Apr 2013
		US 2014302934 A9	09 Oct 2014
		US 8882599 B2	11 Nov 2014
		US 2014101184 A1	10 Apr 2014
		US 8886711 B2	11 Nov 2014
		US 2014012899 A1	09 Jan 2014

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 8892698 B2	18 Nov 2014
		US 2012102316 A1	26 Apr 2012
		US 8909858 B2	09 Dec 2014
		US 2014214909 A1	31 Jul 2014
		US 8914609 B2	16 Dec 2014
		US 2012054500 A1	01 Mar 2012
		US 8914669 B2	16 Dec 2014
		US 2014108493 A1	17 Apr 2014
		US 8914707 B2	16 Dec 2014
		US 2011078377 A1	31 Mar 2011
		US 8924387 B2	30 Dec 2014
		US 2011055473 A1	03 Mar 2011
		US 2014372695 A9	18 Dec 2014
		US 8938591 B2	20 Jan 2015
		US 2010217796 A1	26 Aug 2010
		US 8965956 B2	24 Feb 2015
		US 2012030437 A1	02 Feb 2012
		US 8966194 B2	24 Feb 2015
		US 2013283095 A1	24 Oct 2013
		US 9009575 B2	14 Apr 2015
		US 2011107036 A1	05 May 2011
		US 9015431 B2	21 Apr 2015
		US 2011289283 A1	24 Nov 2011
		US 9026758 B2	05 May 2015
		US 2011125999 A1	26 May 2011
		US 9027080 B2	05 May 2015
		US 2014280681 A1	18 Sep 2014
		US 9063658 B2	23 Jun 2015
		US 2013304711 A1	14 Nov 2013
		US 9063968 B2	23 Jun 2015
		US 2013232307 A1	05 Sep 2013

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 9092140 B2	28 Jul 2015
		US 2011213940 A1	01 Sep 2011
		US 9092439 B2	28 Jul 2015
		US 2014115334 A1	24 Apr 2014
		US 9094402 B2	28 Jul 2015
		US 2014281804 A1	18 Sep 2014
		US 9098376 B2	04 Aug 2015
		US 2009094250 A1	09 Apr 2009
		US 2011055178 A1	03 Mar 2011
		US 2011055273 A1	03 Mar 2011
		US 2011055474 A1	03 Mar 2011
		US 2011072115 A1	24 Mar 2011
		US 2011107078 A1	05 May 2011
		US 2011107113 A1	05 May 2011
		US 2013305060 A9	14 Nov 2013
		US 2011161754 A1	30 Jun 2011
		US 2011286595 A1	24 Nov 2011
		US 2012137091 A1	31 May 2012
		US 2012137095 A1	31 May 2012
		US 2014026017 A1	23 Jan 2014
		US 2014075134 A1	13 Mar 2014
		US 2014122970 A1	01 May 2014
		US 2014123244 A1	01 May 2014
		US 2014164551 A1	12 Jun 2014
		US 2014181578 A1	26 Jun 2014
		US 2014281223 A1	18 Sep 2014
		US 2014298085 A1	02 Oct 2014
		US 2014298135 A1	02 Oct 2014
		US 2014298136 A1	02 Oct 2014
		US 2014310310 A1	16 Oct 2014
		US 2014310491 A1	16 Oct 2014

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 2014310492 A1	16 Oct 2014
		US 2014310775 A1	16 Oct 2014
		US 2014317224 A1	23 Oct 2014
		US 2014325260 A1	30 Oct 2014
		US 2014325305 A1	30 Oct 2014
		US 2014330923 A1	06 Nov 2014
		US 2014331085 A1	06 Nov 2014
		US 2014344228 A1	20 Nov 2014
		US 2014344284 A1	20 Nov 2014
		US 2014344394 A1	20 Nov 2014
		US 2014351528 A1	27 Nov 2014
		US 2014379675 A1	25 Dec 2014
		US 2015043732 A1	12 Feb 2015
		US 2015074220 A1	12 Mar 2015
		US 2015172386 A1	18 Jun 2015
		US 2015212887 A1	30 Jul 2015
		WO 2007041235 A2	12 Apr 2007
		WO 2007041235 A3	29 Nov 2007
		WO 2007120428 A2	25 Oct 2007
		WO 2007120428 A3	12 Sep 2008
		WO 2007120429 A2	25 Oct 2007
		WO 2007120429 A3	21 Aug 2008
		WO 2007120437 A2	25 Oct 2007
		WO 2007120437 A3	02 Apr 2009
		WO 2009048726 A1	16 Apr 2009
		WO 2009048727 A1	16 Apr 2009
		WO 2009048728 A1	16 Apr 2009
		WO 2009048729 A1	16 Apr 2009
		WO 2009123865 A2	08 Oct 2009
		WO 2009123865 A3	30 Dec 2009
		WO 2010009008 A1	21 Jan 2010

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		WO 2010009009 A1	21 Jan 2010
		WO 2011014437 A2	03 Feb 2011
		WO 2011014437 A3	31 Oct 2013
US 2011219100 A1	08 Sep 2011	US 2011219100 A1	08 Sep 2011
		US 8694668 B2	08 Apr 2014
		EP 2008185 A2	31 Dec 2008
		EP 2008185 A4	05 May 2010
		EP 2201460 A1	30 Jun 2010
		EP 2201460 A4	22 Aug 2012
		EP 2201461 A1	30 Jun 2010
		EP 2201461 A4	22 Aug 2012
		EP 2201469 A1	30 Jun 2010
		EP 2201469 A4	15 Jul 2015
		EP 2260387 A2	15 Dec 2010
		EP 2460080 A2	06 Jun 2012
		JP 2009533759 A	17 Sep 2009
		US 2007079082 A1	05 Apr 2007
		US 7546427 B2	09 Jun 2009
		US 2007174192 A1	26 Jul 2007
		US 7574570 B2	11 Aug 2009
		US 2007079083 A1	05 Apr 2007
		US 7574579 B2	11 Aug 2009
		US 2009254720 A1	08 Oct 2009
		US 7818518 B2	19 Oct 2010
		US 2009094251 A1	09 Apr 2009
		US 7904475 B2	08 Mar 2011
		US 2010179966 A1	15 Jul 2010
		US 7953771 B2	31 May 2011
		US 2007079081 A1	05 Apr 2007
		US 7953937 B2	31 May 2011

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 2010017531 A1	21 Jan 2010
		US 2011125916 A9	26 May 2011
		US 7962641 B1	14 Jun 2011
		US 2010023529 A1	28 Jan 2010
		US 8019960 B2	13 Sep 2011
		US 2010063911 A1	11 Mar 2010
		US 8140777 B2	20 Mar 2012
		US 2009094318 A1	09 Apr 2009
		US 8171101 B2	01 May 2012
		US 2010115063 A1	06 May 2010
		US 8171102 B2	01 May 2012
		US 2010169415 A1	01 Jul 2010
		US 8185614 B2	22 May 2012
		US 2011202568 A1	18 Aug 2011
		US 8190662 B2	29 May 2012
		US 2010250751 A1	30 Sep 2010
		US 8200788 B2	12 Jun 2012
		US 2009094320 A1	09 Apr 2009
		US 8209363 B2	26 Jun 2012
		US 2010023710 A1	28 Jan 2010
		US 8244991 B2	14 Aug 2012
		US 2010023524 A1	28 Jan 2010
		US 8285878 B2	09 Oct 2012
		US 2010306578 A1	02 Dec 2010
		US 8352782 B2	08 Jan 2013
		US 2011077086 A1	31 Mar 2011
		US 8357048 B2	22 Jan 2013
		US 2011289383 A1	24 Nov 2011
		US 8448044 B2	21 May 2013
		US 2012290873 A1	15 Nov 2012
		US 8464096 B2	11 Jun 2013

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 2012246421 A1	27 Sep 2012
		US 8468311 B2	18 Jun 2013
		US 2010169500 A1	01 Jul 2010
		US 8478865 B2	02 Jul 2013
		US 2011107165 A1	05 May 2011
		US 8479078 B2	02 Jul 2013
		US 2011029836 A1	03 Feb 2011
		US 8489915 B2	16 Jul 2013
		US 2011107094 A1	05 May 2011
		US 8522022 B2	27 Aug 2013
		US 2010169391 A1	01 Jul 2010
		US 8533256 B2	10 Sep 2013
		US 2011072210 A1	24 Mar 2011
		US 8549351 B2	01 Oct 2013
		US 2011029809 A1	03 Feb 2011
		US 8555109 B2	08 Oct 2013
		US 2010161916 A1	24 Jun 2010
		US 8560882 B2	15 Oct 2013
		US 2011072321 A1	24 Mar 2011
		US 8572429 B2	29 Oct 2013
		US 2011107026 A1	05 May 2011
		US 8589637 B2	19 Nov 2013
		US 2011029731 A1	03 Feb 2011
		US 8595435 B2	26 Nov 2013
		US 2011289565 A1	24 Nov 2011
		US 8621580 B2	31 Dec 2013
		US 2012030736 A1	02 Feb 2012
		US 8627114 B2	07 Jan 2014
		US 2013254624 A1	26 Sep 2013
		US 8627177 B2	07 Jan 2014
		US 2010287200 A1	11 Nov 2010

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 8630987 B2	14 Jan 2014
		US 2013290809 A1	31 Oct 2013
		US 8631303 B2	14 Jan 2014
		US 2012198203 A1	02 Aug 2012
		US 8688949 B2	01 Apr 2014
		US 2012198197 A1	02 Aug 2012
		US 8694752 B2	08 Apr 2014
		US 2011029711 A1	03 Feb 2011
		US 8706980 B2	22 Apr 2014
		US 2013275833 A1	17 Oct 2013
		US 8713358 B2	29 Apr 2014
		US 2011106855 A1	05 May 2011
		US 8732206 B2	20 May 2014
		US 2011106904 A1	05 May 2011
		US 8769035 B2	01 Jul 2014
		US 2011016122 A1	20 Jan 2011
		US 8819011 B2	26 Aug 2014
		US 2011071988 A1	24 Mar 2011
		US 8819179 B2	26 Aug 2014
		US 2014040662 A1	06 Feb 2014
		US 8819482 B2	26 Aug 2014
		US 2013275834 A1	17 Oct 2013
		US 8819516 B2	26 Aug 2014
		US 2014053018 A1	20 Feb 2014
		US 8826067 B2	02 Sep 2014
		US 2012027134 A1	02 Feb 2012
		US 8842746 B2	23 Sep 2014
		US 2011161681 A1	30 Jun 2011
		US 8856552 B2	07 Oct 2014
		US 2011286594 A1	24 Nov 2011
		US 8861727 B2	14 Oct 2014

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 2013275844 A1	17 Oct 2013
		US 8868969 B2	21 Oct 2014
		US 2011289366 A1	24 Nov 2011
		US 8874868 B2	28 Oct 2014
		US 2008183975 A1	31 Jul 2008
		US 8880799 B2	04 Nov 2014
		US 2014032961 A1	30 Jan 2014
		US 8880940 B2	04 Nov 2014
		US 2013102395 A1	25 Apr 2013
		US 2014302934 A9	09 Oct 2014
		US 8882599 B2	11 Nov 2014
		US 2014101184 A1	10 Apr 2014
		US 8886711 B2	11 Nov 2014
		US 2014012899 A1	09 Jan 2014
		US 8892698 B2	18 Nov 2014
		US 2012102316 A1	26 Apr 2012
		US 8909858 B2	09 Dec 2014
		US 2014214909 A1	31 Jul 2014
		US 8914609 B2	16 Dec 2014
		US 2012054500 A1	01 Mar 2012
		US 8914669 B2	16 Dec 2014
		US 2014108493 A1	17 Apr 2014
		US 8914707 B2	16 Dec 2014
		US 2011078377 A1	31 Mar 2011
		US 8924387 B2	30 Dec 2014
		US 2011055473 A1	03 Mar 2011
		US 2014372695 A9	18 Dec 2014
		US 8938591 B2	20 Jan 2015
		US 2010217796 A1	26 Aug 2010
		US 8965956 B2	24 Feb 2015
		US 2012030437 A1	02 Feb 2012

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 8966194 B2	24 Feb 2015
		US 2013283095 A1	24 Oct 2013
		US 9009575 B2	14 Apr 2015
		US 2011107036 A1	05 May 2011
		US 9015431 B2	21 Apr 2015
		US 2011289283 A1	24 Nov 2011
		US 9026758 B2	05 May 2015
		US 2011125999 A1	26 May 2011
		US 9027080 B2	05 May 2015
		US 2014280681 A1	18 Sep 2014
		US 9063658 B2	23 Jun 2015
		US 2013304711 A1	14 Nov 2013
		US 9063968 B2	23 Jun 2015
		US 2013232307 A1	05 Sep 2013
		US 9092140 B2	28 Jul 2015
		US 2011213940 A1	01 Sep 2011
		US 9092439 B2	28 Jul 2015
		US 2014115334 A1	24 Apr 2014
		US 9094402 B2	28 Jul 2015
		US 2014281804 A1	18 Sep 2014
		US 9098376 B2	04 Aug 2015
		US 2009094250 A1	09 Apr 2009
		US 2011055178 A1	03 Mar 2011
		US 2011055273 A1	03 Mar 2011
		US 2011055474 A1	03 Mar 2011
		US 2011072115 A1	24 Mar 2011
		US 2011107078 A1	05 May 2011
		US 2011107113 A1	05 May 2011
		US 2013305060 A9	14 Nov 2013
		US 2011161754 A1	30 Jun 2011
		US 2011286595 A1	24 Nov 2011

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		US 2012137091 A1	31 May 2012
		US 2012137095 A1	31 May 2012
		US 2014026017 A1	23 Jan 2014
		US 2014075134 A1	13 Mar 2014
		US 2014122970 A1	01 May 2014
		US 2014123244 A1	01 May 2014
		US 2014164551 A1	12 Jun 2014
		US 2014181578 A1	26 Jun 2014
		US 2014281223 A1	18 Sep 2014
		US 2014298085 A1	02 Oct 2014
		US 2014298135 A1	02 Oct 2014
		US 2014298136 A1	02 Oct 2014
		US 2014310310 A1	16 Oct 2014
		US 2014310491 A1	16 Oct 2014
		US 2014310492 A1	16 Oct 2014
		US 2014310775 A1	16 Oct 2014
		US 2014317224 A1	23 Oct 2014
		US 2014325260 A1	30 Oct 2014
		US 2014325305 A1	30 Oct 2014
		US 2014330923 A1	06 Nov 2014
		US 2014331085 A1	06 Nov 2014
		US 2014344228 A1	20 Nov 2014
		US 2014344284 A1	20 Nov 2014
		US 2014344394 A1	20 Nov 2014
		US 2014351528 A1	27 Nov 2014
		US 2014379675 A1	25 Dec 2014
		US 2015043732 A1	12 Feb 2015
		US 2015074220 A1	12 Mar 2015
		US 2015172386 A1	18 Jun 2015
		US 2015212887 A1	30 Jul 2015
		WO 2007041235 A2	12 Apr 2007

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2015/030163

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
		WO 2007041235 A3	29 Nov 2007
		WO 2007120428 A2	25 Oct 2007
		WO 2007120428 A3	12 Sep 2008
		WO 2007120429 A2	25 Oct 2007
		WO 2007120429 A3	21 Aug 2008
		WO 2007120437 A2	25 Oct 2007
		WO 2007120437 A3	02 Apr 2009
		WO 2009048726 A1	16 Apr 2009
		WO 2009048727 A1	16 Apr 2009
		WO 2009048728 A1	16 Apr 2009
		WO 2009048729 A1	16 Apr 2009
		WO 2009123865 A2	08 Oct 2009
		WO 2009123865 A3	30 Dec 2009
		WO 2010009008 A1	21 Jan 2010
		WO 2010009009 A1	21 Jan 2010
		WO 2011014437 A2	03 Feb 2011
		WO 2011014437 A3	31 Oct 2013
WO 2014005279 A1	09 Jan 2014	WO 2014005279 A1	09 Jan 2014