SYSTEM AND METHODS FOR DISTRIBUTED DATA STORAGE

FIELD OF THE INVENTION

[0001] The present invention generally relates to data storage, and more particularly though not exclusively relates to systems and methods for non-homogeneous distributed data storage, non-maximum distance separable (MDS) distributed data storage, and locally repairable codes.

BACKGROUND OF THE DISCLOSURE

[0002] Cloud storage or distributed storage systems (DSS) are becoming more popular because they allow users to access the stored information from anywhere. Since the information is being stored at multiple remote servers, it is safe as it is not subject to a single point of failure as compared to local storage. Although local storage cost is inexpensive, to store equivalent amounts of data in the cloud or at a data centre can be expensive. The higher cost is typically due to the communication bandwidth and the reliability built into the system to ensure that it is rarely subject to failures due to natural disasters, hardware failures, or power blackout. Besides requiring low storage cost and high security, a DSS needs to be robust such that when a node fails, it can be repaired within a short period of time. In addition, based on data content, there are various data storage requirements as summarized in Table 1.

Content type	**Data Size**	**Data Update Freq**	**Data Access Freq**
Data	Small	High	High
Multimedia	Medium	Low	Medium
Backup	Large	Medium	Low

Table 1

[0003] Thus, what is needed are highly recoverable and relatively impervious to failure data storage methods and systems for distributed storage systems. Furthermore, other

desirable features and characteristics will become apparent from the subsequent detailed description, taken in conjunction with the accompanying drawings and this background of the disclosure.

SUMMARY

[0004] In general terms in a first aspect the invention proposes a non-homogeneously distributed data storage. In a second aspect the invention proposes a distributed data storage using repair pairs, XOR based coding and/or non MDS coding. In a third aspect the invention proposes locally repairable codes for a range of coding parameters where the field size is minimised.

[0005] In a first specific expression of the invention there is provided a systematic distributed storage system (DSS) comprising

a plurality of storage nodes, wherein each storage node configures to store a plurality of sub-blocks of a data file and a plurality of coded blocks; and

a set of repair pairs for each of the storage nodes;

wherein the system is configured to use the respective repair pair of storage nodes to repair a lost or damaged sub-block or coded block on a given storage node.

[0006] In a second specific expression of the invention there is provided a distributed storage system DSS comprising

h non-empty nodes; and

data stored non-homogenously across the non-empty nodes according to the storing codes (n,k).

[0007] In a second specific expression of the invention there is provided a method for determining linear erasure codes with local repairability comprising

selecting two or more coding parameters including r and d;

determining if an optimal $[n, k, d]$ code having all-symbol (r, d)-locality ("$(r, \mathrm{d})_a$") exists for the selected r, d; and

if the optimal $(r, d)_a$ code exists performing a local repairable code using the optimal $(r, d)_a$ code.

[0008] One or more embodiments may be implemented according to any of claims 2 to 7, 9 to 18 and 20 to 25.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The invention will now be described, by way of example only, with reference to the figures, of which:

Figures 1A and 1B are schematic diagrams of an architecture for a DSS system;

Figure 2 is a schematic diagram of a typical encoding structure for DSS;

Figure 3 is a flow diagram of the selection of DSS system parameters based on content and encoding scheme;

Figure 4 is a schematic diagram of the encoding process for each data block;

Figure 5 is a schematic diagram of the repair process when one node fails;

Figure 6 is a schematic diagram of the repair process when one node fails;

Figure 7 is a schematic diagram of 1 node failure repair using scheme A based on (5, 3) MDS codes in non-homogeneous distributed storage systems;

Figure 8 is a schematic diagram of 1 node failure repair, where the total repair bandwidth is $M/2$ and is smaller than the bound;

Figure 9 is a schematic diagram of 1 node failure repair using scheme B based on (5, 3) MDS codes in non-homogeneous distributed storage systems;

Figure 10A is a schematic diagram of 2 nodes failure repair using scheme A in non-homogeneous distributed storage system;

Figure 10B is a schematic diagram of 2 nodes failure repair using scheme C in non-homogeneous distributed storage system;

Figure 11 is a schematic diagram of data allocation using (8,5) MDS code in homogeneous DSS and non-homogeneous DSS;

Figure 12 is a graph comparing of data availability between super-node non-homogeneous DSS and homogeneous DSS;

Figure 13 is a schematic diagram of repairing failure when n = h(n - k) in the (n = 6,k = 4,h = 3) non-homogeneous DSS;

Figure 14. An example of repair multi-failures based on (n = 8,k = 5,r = 2) MSR codes using 3 storage nodes;

Figure 15. A comparison of data availability between minimum-spread non-homogeneous DSS and homogeneous DSS;

Figure 16 is a schematic diagram of how a locally repairable linear code is used to construct a distributed storage system: a file F is first split into five packets of equal size $\{x_1,\cdots,x_5\}$ and then is encoded into 12 packets, using a $(2,3)_a$ linear code. These 12 encoded packets are stored at 12 nodes $\{v_1,\cdots,v_{12}\}$, which are divided into three groups $\{v_1,v_2,v_3,v_4\}$, $\{v_5,v_6,v_7,v_8\}$ and $\{v_9,v_{10},v_{11},v_{12}\}$. Each group can perform local repair of up to two node failures. For example, if node v_9 fails, it can be repaired by any two packets among v_{10}, v_{11} and v_{12}. Moreover, the entire file F can be recovered by five packets from any five nodes $v_{i1},\cdots,v_{i5}$ which intersect each group with at most two packets. For example, F can be recovered from five packets stored at v_1, v_3, v_7, v_8 and v_{10};

Figure 17 is a schematic diagram of optimal $(r,d)_a$ linear codes;

Figure 18 is an (A; ψ)-frame, where $n = 37$; $r = d = 3$; $t = 8$; $A_1 = \{1; 2; 3\}$, $A_2 = \{4; 5\}$;$B = \{6; 7; 8\}$; $\mathcal{A} = \{A_1;A_2;B\}$ and $\psi = \{1; 14\}$.

DETAILED DESCRIPTION

[0010] The following detailed description is merely exemplary in nature and is not intended to limit the invention or the application and uses of the invention. Furthermore, there is no intention to be bound by any theory presented in the preceding background of the invention or the following detailed description of the invention.

[0011] The following definitions will be used through the description:

M	block size
m	number of blocks
q	field size
k	number of sub-blocks for each block of size M,
n	number of coded blocks, also number of nodes
C	level of redundancy (number of tolerable node failures)
d	repair degree
F_2	the finite field of two elements
r	the parameter that determines C, $C = 2^r - 1$
k'	$k' = k/r$
z	coded blocks
o	original blocks
j	$1 = j = n$, index of the coded blocks
i	$1 = i = k'$, one index of the original blocks
l	running index in the sum
x	systematic coded blocks
t	number of node failures
s	systematic node
p	parity node
h	non-empty nodes
v	repair matrix
T	used on the right-upper of a matrix to denote the transpose of it
w	node's weight
y	number of downloaded blocks from node i
p	node's online probability
a	storage size at node i
d	update bandwidth
G	a generating matrix of a linear code
O	an index set used in a certain round of a process
Λ	the set of subsets of $\{1,2,\ldots,n\}$ that satisfy a certain properties.
γ	repair bandwidth of failure node i
f	block i divided from file of size M
β	downloaded packet from node i
A	one possible combinations of online nodes
F_q	the finite field of q elements.
S	a subset of the set $\{1,2,\ldots,n\}$

[0012] The present embodiment proposes methodologies, namely Non-MDS DSS or XOR based DSS, Non-Homogeneous DSS, and Locally repairable codes. XOR based DSS is best suitable for data storage and peer-to-peer backup system, while Non-Homogeneous and Locally repairable DSS is best suitable for backup system. Table 2 summarizes the applicability of these two schemes to various data content.

Content type	Data Size	Data Update Freq	Data Access Freq	Proposed Technology

Data	Small	High	High	Non-MDS DSS
Multimedia	Medium	Low	Medium	
Backup	Large	Medium	Low	Non-Homogenous DSS Locally Repairable Code
Backup (peer-to-peer)				Non-MDS DSS

Table 2

[0013] In accordance with the present embodiment, two DSS architectures are presented in FIGs. 1A and 1B. In FIG. 1A, a controller centric architecture is depicted. In this architecture, the client only deals with the controller and the controller will distribute, store, and retrieve information on behalf of the client. Referring to FIG. 1B, a client centric architecture is presented. In this architecture, the controller gives the client information about the distributed storage servers and client stores and retrieves the information directly to/from the distributed storage servers. The same architecture used for distributing, storing, and retrieving information can be applied when repairing a failed storage node in the DSS. Operation in accordance with the present embodiment can be implemented in either of the two architectures.

[0014] FIG. 2 depicts a typical DSS encoding structure. An information/data block is divided into m blocks of size M each ($m = 1$). The size of mM could be more than the size of the information block due to some constraints imposed on M for some encoding schemes. Each block of size M is further divided into k sub-blocks, and these k sub-blocks are encoded into n coded blocks of encoded data to be stored on n distributed storage servers, termed "nodes".

[0015] Referring to FIG. 3, a methodology to determine the size for m and M in accordance with one aspect of the present embodiment is depicted. The size of m and M can be determined based on the encoding scheme, the storage server bandwidth, and the content type of the information/data block. For example, for photographic and audio data,

m can be set as 1, and M can be rounded to the nearest integer. Hence, the allowable value for M within an encoding scheme is preferably small.

[0016] Hereinafter, three architectures for DSS which permit dynamic selection of file fragment size in response to encoding scheme, storage server bandwidth, and file content type in accordance with the present embodiment will be discussed: XOR based DSS, Non-Homogeneous DSS, and Locally Repairable DSS.

XOR based distributed storage with binary Simplex code

[0017] In a data centre, there may be three requirements that need to be satisfied. First, when the user wants to retrieve the original data, he should be able to obtain it as soon as possible. This requirement may be important. Next, the update complexity of the code should be as low as possible because the user may modify the data frequently. And last, minimization of storage space used so as to reduce the cost of energy consumption and the number of devices, while still tolerating a certain number of node failures.

[0018] Four types of redundancy schemes can be considered to implement this aspect of the present embodiment: (1) Replication, (2) Reed-Solomon codes, (3) Regenerating codes, and (4) Self-Repairing codes. Replication is the de facto standard for redundancy implementations, but it has a large storage cost. For example, if a system is to tolerate C node failures, C copies of the original data need to be stored. Both Reed-Solomon codes and regenerating codes are maximum distance separable (MDS) codes. In other words, when a file (or a portion of file) consisting of k blocks is encoded into n coded blocks and each coded block is stored at a unique physical node, connecting to any k nodes can recover the original file and the system can tolerate any $n - k$ nodes failure. However, Reed-Solomon codes and regenerating codes both have a high update complexity. In addition, given the fact that the decoding is done over a larger field, these two codes together with self-repairing codes have high decoding complexity when the user retrieves the original data. Hence, Reed-Solomon codes, regenerating codes and self-repairing

codes cannot guarantee the first two requirements set out above, making them unfit for data centre application.

[0019] In accordance with the present embodiment, a type of Non-MDS code with repair degree $d = 2$ over F_2 can be constructed as follows: If the system is designed to tolerate $C = 2^{r-1} - 1$ nodes failure, r is determined once C is specified; then a file (or a portion of file) of size M is divided into $k = k'r$ sub-blocks $o_{1,1}, \cdots, o_{1,r}$, $o_{2,1}, \cdots, o_{2,r}$, $o_{k',1}, \cdots, o_{k',r}$; next the k sub-blocks of information are linearly encoded into $n = k'(2^r - 1)$ coded blocks over F_2, denoted as $z_1, z_2, \cdots, z_n$, with each coded block being stored at a unique physical node; and, finally, all the n coded blocks, where $n = k' \cdot (2^r - 1)$, in the system are constructed as follows:

$$z_j = \begin{pmatrix} \alpha_{j,1} & \alpha_{j,2} & \cdots & \alpha_{j,r} \end{pmatrix} \begin{pmatrix} o_{i,1} \\ o_{i,2} \\ \vdots \\ o_{i,r} \end{pmatrix} = \sum_{l=1}^{r} \alpha_{j,l} o_{i,l}, (1 \leq j \leq n) \qquad (1)$$

where $i = \left\lfloor \frac{j-1}{2^r-1} \right\rfloor + 1$, $\alpha_{j,l} \in F_2 (1 \leq l \leq \mathrm{r})$, and $\begin{pmatrix} \alpha_{j,1} & \alpha_{j,2} & \cdots & \alpha_{j,r} \end{pmatrix}$ is the binary representation of $j - \left\lfloor \frac{j-1}{2^r-1} \right\rfloor (2^r - 1)$ and $\lfloor\ \rfloor$ represents the integer floor. The constructed code has a minimum repair degree d of 2. If a node fails, a newcomer can obtain the lost information in the failed node by only connecting and downloading two coded blocks from two surviving nodes. Such two surviving nodes are called a repair pair. Each node can find at least C repair pairs for repairing it.

[0020] FIGs. 4 and 5 show this model graphically. When node l fails, the newcomer downloads two coded blocks z_i and z_j from selected node i and j, where z_l can be recovered from z_i and z_j. In this system, the newcomer can find C such repair pairs. Thus, it can be seen that Non-MDS code with repair degree $d = 2$ over F_2 has the following features: (a) Due to the encoding over F_2, this coding can be implemented with a XOR operation which is computationally efficient, and the user can decode the original data rapidly after downloading the necessary data; (b) The repair degree is $d = 2$, which means

if a node fails, a newcomer can obtain the lost block from the failed node by connecting to only two surviving nodes, such two surviving nodes being called a repair pair; (c) The system can tolerate C nodes of failure, where C is a design parameter; (d) Low update complexity is provided because, if an original block is changed, only $C + 1$ nodes in the system need to be updated (this low update complexity is roughly equivalent to the update complexity of replication); and (e) Every node has at least C repair pairs for repairing it.

[0021] In system in accordance with the present embodiment, in order to satisfy the following criteria, $n \geq k'(2^r - 1)$: (1) There must exist k linearly independent coded blocks $x_1, x_2, \cdots, x_k$ selected from $z_1, z_2, \cdots, z_n$ in the system in order to recover $o_1, o_2, \cdots, o_k$; and (2) Each node can find $C = 2^{r-1} - 1$ ways for repairing it. The code in accordance with the present embodiment achieves the above minimum n satisfying the two criteria for the system.

[0022] A specific example will now be given for the code in accordance with the present embodiment. Consider a Non-MDS codes with repair degree $d = 2$ over F_2 wherein the original file is divided into $k = k' \times r = 2 \times 3 = 6$ blocks: $o_{1,1}, o_{1,2}, o_{1,3}, o_{2,1}, o_{2,2}, o_{2,3}$. A set is then provided as set out in Equation 2:

$$\mathbf{A}_i = \begin{pmatrix} \alpha_{i(2^r-1)+1,1} & \alpha_{i(2^r-1)+1,2} & \cdots & \alpha_{i(2^r-1)+1,r} \\ \alpha_{i(2^r-1)+2,1} & \alpha_{i(2^r-1)+2,2} & \cdots & \alpha_{i(2^r-1)+2,r} \\ & \vdots & & \\ \alpha_{(i+1)(2^r-1),1} & \alpha_{(i+1)(2^r-1),2} & \cdots & \alpha_{(i+1)(2^r-1),r} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad (2)$$

where $0 \leq i < k'$. The $n = k'(2^r - 1) = 14$ coded blocks in the system are encoded as shown in Equation 3:

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{pmatrix} = \mathbf{A}_0 \begin{pmatrix} o_{1,1} \\ o_{1,2} \\ o_{1,3} \end{pmatrix}, \quad \begin{pmatrix} z_8 \\ z_9 \\ z_{10} \\ z_{11} \\ z_{12} \\ z_{13} \\ z_{14} \end{pmatrix} = \mathbf{A}_1 \begin{pmatrix} o_{2,1} \\ o_{2,2} \\ o_{2,3} \end{pmatrix} \qquad (3)$$

[0023] In accordance with this system, one code failure can be repaired by connecting to two surviving nodes and the number of repair pairs is $C = 2^{3-1} - 1 = 3$. For example, if z_1 is lost, it can be repaired by using (z_2, z_3), (z_4, z_5) or (z_6, z_7). Table 3 summarizes the repair pairs for all possible failures:

	1st Repair Pair	2nd Repair Pair	3rd Repair Pair
z_1	(z_2, z_3)	(z_4, z_5)	(z_6, z_7)
z_2	(z_1, z_3)	(z_4, z_6)	(z_5, z_7)
z_3	(z_1, z_2)	(z_5, z_6)	(z_4, z_7)
z_4	(z_1, z_5)	(z_2, z_6)	(z_3, z_7)
z_5	(z_1, z_4)	(z_3, z_6)	(z_2, z_7)
z_6	(z_2, z_4)	(z_3, z_5)	(z_1, z_7)
z_7	(z_1, z_6)	(z_2, z_5)	(z_3, z_4)

Table 3

[0024] In regards to codes for the present embodiment, we compare replication, Reed-Solomon codes, Exact Minimum Storage Regenerating Codes (E-MSR), and self-repairing codes from four aspects: (1) Update complexity; (2) Complexity for retrieving the original data; (3) Storage efficiency; (4) Repair bandwidth

[0025] In accordance with the present embodiment, when an original block needs to be changed, advantageously only $C + 1$ nodes in the system need to be modified. This update complexity of the proposed codes is the same as replication. Reed-Solomon codes and E-MSR without systematic codes both need to update all the nodes in the system. E-MSR with systematic codes needs to update n-k+1 nodes in the system, while self-repairing codes need to update $2C + 1$ codes in the system.

[0026] In accordance with the present embodiment, when a user wants to retrieve the original k sub-blocks and the systematic coded blocks $x_{1,1}, \cdots, x_{1,r}$, $x_{2,1}, \cdots, x_{2,r}$, $x_{k',1}, \cdots, x_{k',r}$ are available, the user can download the original k blocks directly; if the

systematic coded blocks are not available, decoding to retrieve the original data can be very fast due to the computational efficiency of XOR operation.

[0027] Replication can download the k original data sub-blocks directly. Reed-Solomon codes, regenerating codes without systematic codes, and self-repairing codes however need to perform a decoding operation over a field whose size is larger than 2, making them unfit for the case where users want to retrieve the original data in a real-time manner. For regenerating codes with systematic codes in the system, when the user wants to retrieve the original data, downloading efficiency is similar to the present embodiment when the systematic codes are available.

[0028] Further, for replication, self-repairing codes and the present embodiment, the user must choose k nodes selectively to retrieve the original data, while Reed-Solomon codes and regenerating codes allow selection of an arbitrary k nodes in the system.

[0029] If the system can tolerate $C = 2^{r-1} - 1$ node failures, operation in accordance with the present embodiment and self-repairing codes need to store $\frac{2^r-1}{r}$ times of the original data. Replication needs to store $2^{r-1} - 1$ times of the original data. Reed-Solomon codes and regenerating codes need to store $\frac{k+C}{k}$ times of the original data.

[0030] For one node failure, the repair bandwidth for both the present embodiment and self-repairing codes is $\frac{2}{k}M$, where M is the size of the data block that the encoding scheme is applied to. The repair bandwidth for replication is $\frac{M}{k}$. Reed-Solomon codes need to download data of size M, while regenerating codes need to download data of size $\frac{M}{k} \cdot \frac{d}{d-k+1}$, where d is the number of nodes connected to complete repair.

[0031] For $t, (t \geq 2)$ node failures, the repair bandwidth of R-S codes, self-repairing codes, E-MSR with $n - t \geq d$, and the present embodiment is t times their respective repair bandwidths. For E-MSR with $n - t < d$, the repair bandwidth is t·M.

[0032] The above comparisons are summarized in Table 4:

	Field	Update Complexity	Retrieving the original data	Storage Cost (n)	Repair bandwidth (1 node failure)	Repair bandwidth (t node failures, $t \geq 2$)
R-S code	F_q $(q \geq n+1)$	Update all n nodes	Solving a $k \times k$ linear equation over F_q	$k + 2^{r-1} + 1$	M	$2M$
E-MSR without systematic codes	F_q^*				$\frac{M \cdot d}{k(d-k+1)}$ $(d \geq 2k-2)$	$\frac{t \cdot M \cdot d}{k(d-k+1)}$ (if $n-t \geq d$)
E-MSR with systematic codes		Update $n-k+1$ nodes	Directly			$t \cdot M$ (if $n-2 < d$)
Self-repairing code	$F_{q^{M/k}}$ $(M/k \geq k)$	Update $2C+1$ nodes	Solving a $k \times k$ linear equation over F_q	$k'(2^r-1)$	$2\frac{M}{k}$	$\frac{2 \cdot t \cdot M}{k}$
Replication	Nil	Update $C+1$ nodes	Directly	$k(2^{r-1}-1)$	$\frac{M}{k}$	$\frac{t \cdot M}{k}$
The present embodiment	F_2	Update $C+1$ nodes	Directly	$k'(2^r-1)$	$2\frac{M}{k}$	$\frac{2 \cdot t \cdot M}{k}$

Table 4 (**Note**: *q may be $2k$+3, $2n$ and n^2 etc. depending on the specific schemes.)

[0033] A specific example of the above comparison is given in Table 5. In this case, we set fault-tolerance ability $C = 2^4 - 1 = 15$ and $k = 3 \times 5 = 15$. Here, $r = 5$ and $k' = 3$. For E-MSR with systematic codes and E-MSR without systematic codes, we adopt the conventional schemes.

	Field	Update Complexity	Retrieving the original data	Storage Cost	Repair bandwidth (1 node failure)	Repair bandwidth (t node failures, $t \geq 2$)
R-S code	F_{32}	Update all the nodes in the system	Solving a 15×15 linear	30	M	$t \cdot M$
E-MSR without	F_{900}				$\frac{M \cdot d}{15(d-14)}$	$\frac{t \cdot M \cdot d}{15(d-14)}$

systematic code*			equation over F_q			
Self-repairing code	F_{32}	31 nodes		93	$\frac{2}{15}M$	$\frac{2 \cdot t}{15}M$
E-MSR with systematic code**	F_{30}	16 nodes	Directly	30	$\frac{M \cdot d}{15(d-14)}$	$t \cdot M$
Replication	Nil	16 nodes	Directly	225	$\frac{1}{15}M$	$\frac{t}{15}M$
The proposed code	F_2	16 nodes	Directly	93	$\frac{2}{15}M$	$\frac{2 \cdot t}{15}M$

Table 5 **Note:** * d must be no less than $2k-2$.
**d must be no less than $2k-1$.

[0034] To summarize, for an application such as a data centre (e.g., Dropbox™), it is the most important to retrieve the data in a simple way and keep update complexity low as the data will be accessed and updated frequently. In such applications, Reed-Solomon codes, regenerating codes and self-repairing codes fail to satisfy these two requirements. Only replication and non-MDS DSS systems and methods in accordance with the present embodiment are suitable candidates. However, as compared with replication, non-MDS DSS operation in accordance with the present embodiment has much better performance in terms of storage efficiency while providing the same fault-tolerance ability. Higher storage efficiency means that less storage devices and less energy consumption are needed.

Extended Simplex Code

[0035] Now, we propose an extended model of the previous Non-MDS code. The main difference is adding a parity coordinate to the simplex code. The encoding is over $\mathbb{F}_2$ (same as previously). The repair degree is $d = 3$. The system can tolerate C nodes of failure. Note that C must be a power of 2. The update complexity is C (same as previously). Every node has at least $2C-1$ repair pairs for repairing it.

[0036] A type of extended Non-MDS code with repair degree d = 3 over $\mathbb{F}_2$ can be constructed and described as follows. The system is designed to tolerate $C = 2^{r-1}$

failures, r is determined once C is specified. A file of size M is divided into $k = k'r$ blocks. The k blocks of information are linearly encoded into $n = k'2^r$ coded blocks over $\mathbb{F}_2$. The generator matrix E_i of the extended code can be described in terms of the generator matrix of the previous case A_i as shown in Equation 4:

$$E_i = \begin{pmatrix} A_i & \mathbf{1} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} 1\,0\,0\,1 \\ 0\,1\,0\,1 \\ 1\,1\,0\,1 \\ 0\,0\,1\,1 \\ 1\,0\,1\,1 \\ 0\,1\,1\,1 \\ 1\,1\,1\,1 \\ 0\,0\,0\,1 \end{pmatrix} \tag{4}$$

[0037] You just add the all-one column **1** and the row all-zeros **0** except the last one. After adding this rows/columns you should make columns operations to be sure that identity matrix is a sub matrix of E_i in order to create a systematic code. This can be done through Gaussian elimination over columns as shown in Equation 5:

$$\begin{pmatrix} 1\,0\,0\,1 \\ 0\,1\,0\,1 \\ 1\,1\,0\,1 \\ 0\,0\,1\,1 \\ 1\,0\,1\,1 \\ 0\,1\,1\,1 \\ 1\,1\,1\,1 \\ 0\,0\,0\,1 \end{pmatrix} \dashrightarrow \begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 1\,1\,0\,0 \\ 0\,0\,1\,0 \\ 1\,0\,1\,1 \\ 0\,1\,1\,1 \\ 1\,1\,1\,1 \\ 0\,0\,0\,1 \end{pmatrix} \tag{5}$$

[0038] The code has repair degree $d = 3$. Each node can find at least $2^r - 1$ repair triples. Consider a non-MDS code with repair degree $d = 3$ over $\mathbb{F}_2$ as follows: the original file is divided into $k = k'r = 2 \cdot 4 = 8$. You can write the same as the previous case but now the matrix is shown in Equation 6:

$$\begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 1\,1\,0\,0 \\ 0\,0\,1\,0 \\ 1\,0\,1\,1 \\ 0\,1\,1\,1 \\ 1\,1\,1\,1 \\ 0\,0\,0\,1 \end{pmatrix} \tag{6}$$

[0039] The $n = k'2^r = 2 \cdot 8 = 16$. Then, the encoding is to encode 4 information coordinates into 8 as shown in Equation 7:

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \end{pmatrix} = E_0 \begin{pmatrix} o_{1,1} \\ o_{1,2} \\ o_{1,3} \\ o_{1,4} \end{pmatrix}, \begin{pmatrix} z_9 \\ z_{10} \\ z_{11} \\ z_{12} \\ z_{13} \\ z_{14} \\ z_{15} \\ z_{16} \end{pmatrix} = E_1 \begin{pmatrix} o_{2,1} \\ o_{2,2} \\ o_{2,3} \\ o_{2,4} \end{pmatrix} \tag{7}$$

[0040] In this system, one failure can be repaired by connecting three surviving nodes and the available number of such repair triples is $2^3 - 1 = 7$. For example, if z_1 is lost, it can be repaired by using $(z_4,z_5,z_8),(z_2,z_3,z_4),(z_2,z_7,z_8),(z_3,z_5,z_7),(z_2,z_5,z_6),(z_4,z_6,z_7)$ or (z_3,z_6,z_8). The repair triples for all nodes are summarized in Table 6.

	1st repair triple	2nd repair triple	3rd repair triple	4th repair triple	5th repair triple	6th repair triple	7th repair triple
z_1	z_4,z_5,z_8	z_2,z_3,z_4	z_2,z_7,z_8	z_3,z_5,z_7	z_2,z_5,z_6	z_4,z_6,z_7	z_3,z_6,z_8
z_2	z_1,z_3,z_4	z_3,z_6,z_7	z_1,z_7,z_8	z_4,z_5,z_7	z_3,z_5,z_8	z_1,z_5,z_6	z_4,z_6,z_8
z_3	z_1,z_2,z_4	z_3,z_6,z_7	z_1,z_5,z_7	z_2,z_5,z_8	z_4,z_7,z_8	z_1,z_6,z_8	z_4,z_5,z_6
z_4	z_1,z_5,z_8	z_1,z_2,z_3	z_2,z_5,z_7	z_3,z_7,z_8	z_3,z_7,z_8	z_2,z_6,z_8	z_3,z_5,z_6
z_5	z_1,z_4,z_8	z_3,z_4,z_6	z_1,z_3,z_7	z_2,z_4,z_7	z_2,z_3,z_8	z_1,z_2,z_6	z_6,z_7,z_8
z_6	z_1,z_3,z_8	z_1,z_2,z_5	z_1,z_4,z_7	z_5,z_7,z_8	z_2,z_4,z_8	z_3,z_4,z_5	z_2,z_3,z_7
z_7	z_1,z_3,z_5	z_2,z_3,z_6	z_1,z_4,z_6	z_2,z_4,z_5	z_1,z_2,z_8	z_3,z_4,z_8	z_5,z_6,z_8
z_8	z_1,z_2,z_7	z_1,z_4,z_5	z_1,z_3,z_6	z_3,z_4,z_7	z_2,z_4,z_6	$z_5.z_6,z_7$	z_2,z_3,z_5

Table 6

[0041] The storage efficiency is $\frac{2^r}{r+1}$ and Tables 4 and 5 can thus be appended with Table 7 and 8:

	Field	Update Complexity	Retrieving the original data	Storage Cost (n)	Repair bandwidth (1 node failure)	Repair bandwidth (t node failures, $t \geq 2$)
The proposed extended case	F_1	Update C nodes	Directly	$k'2^r$	$3\frac{M}{k}$	$3t\frac{M}{k}$

Table 7

	Field	Update Complexity	Retrieving the original data	Storage Cost	Repair bandwidth (1 node failure)	Repair bandwidth (t node failures, $t \geq 2$)
The proposed extended case	F_2	16 nodes	Directly	96	$\frac{3}{15}M$	$\frac{3t}{15}M$

Table 8

Non-homogeneous distributed storage system (DSS)

[0042] As discussed above, distributed storage systems (DSS) are widely used today for storing data reliably over long periods of time using a distributed collection of storage nodes which may be individually unreliable. Application scenarios include large data centres and peer-to-peer storage systems that use nodes across the Internet for distributed file storage. One of the challenges for DSS is the repair problem: If a node storing a coded piece fails or leaves the system, we need to create a new encoded piece and store it at a new node in order to maintain the same level of reliability, and we need to do it with a minimum repair bandwidth. To solve this problem, a generic framework based on (n, k, α, d, β) regenerating codes has been introduced in the prior art.

[0043] With (n,k) MDS codes, a data file is encoded and distributed to n storage nodes, any k of which can reconstruct the original file. The data file remains intact even though some storage nodes may fail. In case of node failures, we need to regenerate new nodes

(called the newcomers) to repair the lost data of the failed nodes. The newcomers are regenerated by downloading some data from the surviving nodes. The required traffic for repairing single-node failure, called repair-bandwidth, is another metric in measuring the system performance, which is essential in bandwidth-limited storage networks.

[0044] A class of erasure codes, called regenerating codes, was introduced to reduce the repair-bandwidth of failure nodes. Two novel coding schemes have been proposed and named as minimum storage regenerating (MSR) code and minimum bandwidth regenerating (MBR) code which correspond to the best storage efficiency and the minimum repair bandwidth, respectively.

[0045] However, they assume that each node in the DSS is the same such as storage capacity, reliability and communication bandwidth etc. This assumption does not exploit the heterogeneous characteristic of the real world actual systems. In practice, there can be many storage nodes located at different geography location with different connection bandwidth and reliability issues. In such scenario, we may not need to store information on all the nodes, rather to select a few nodes that have the best connection (or some other criteria) to perform the distributed storage.

[0046] Moreover, in traditional homogeneous DSS, data are encoded into n blocks, and we store all the n blocks at n different nodes. However in many cases, we would prefer to have many distributed nodes for easy management, we do require a smaller number of storage nodes with large bandwidth. We study how to make use of existing MSR codes to apply in such systems.

[0047] We investigate how to apply any (n,k) MSR codes, and store them flexibly in a non-homogenous distributed storage system. We show that by allocating the storage across different nodes efficiently, we can lead to lower download time, higher availability, lower repair bandwidth when all the storage nodes have different parameters or characteristics. Depending on the storage nodes characteristics, we propose three data allocate schemes,

namely super-node, partial-homogeneous and minimum-spread. These schemes exploit the different in bandwidth and availability of storage nodes to allocate data efficiently. Since when a node fails in non-homogenous DSS, it corresponds to multi-node failures in homogeneous DSS, it is a challenging task to repair such multi failures with minimum repair bandwidth. In one aspect, we propose a solution for this repair problem and shows that on general, non-homogenous DSS require less complexity and repair bandwidth than the traditional homogeneous DSS.

[0048] Additional aspects of the super-node non-homogeneous DSS, also proposes two schemes for storing data using a $(k + 2, k)$ maximum distance separable (MDS) codes. These new schemes can achieve optimal repair bandwidth $\frac{k+1}{2}\frac{M}{k}$ at smaller finite field q and 4 times smaller fragment M than conventional DSS. Smaller M and q help the non-homogeneous DSS to save update bandwidth more efficiently than traditional DSS. Moreover, one of the schemes can achieve one failure repair bandwidth at $\frac{M}{2k}$ smaller than optimal bandwidth bounds.

Model of traditional homogeneous DSS

[0049] We follow the definition of traditional homogeneous DSS using (n,k,d,α,γ) regenerating codes over finite field F_q. This network has n storage nodes and every k nodes suffice to reconstruct all the data. The size of the file to be stored is M and partitioned into k equal blocks $\mathbf{f}_1, \cdots, \mathbf{f}_k \in \mathbb{F}_q^N$ where $N = \frac{M}{k}$. After encoding them into n coded blocks using an (n,k) maximum distance separable (MDS) code, we store them at n nodes.

[0050] We define the MDS property of a storage code using the notion of data collectors. A storage code where each node contains $\frac{M}{k}$ worth of storage, has the MDS property if a data collector can reconstruct the original file M by connecting to any k out of n storage nodes.

[0051] When a node fails, the data stored therein is recovered by downloading β packets each from any d $(\geq k)$ of the remaining $(n-1)$ nodes (FIG. 6). Therefore, the total repair bandwidth is $\gamma_1 = d\beta$. The number d of nodes that participate in the repair is named as repair degree. There may be an optimal tradeoff between the storage per node, α, and the bandwidth to repair one node, γ_1. We focus on the extreme point where the smallest $\alpha = \frac{M}{k}$ corresponds to a minimum-storage regenerating (MSR) code as shown in Equation 8:

$$(\alpha, \gamma_1) = \left(\frac{M}{k}, \frac{Md}{k(d-k+1)}\right) \tag{8}$$

[0052] For such minimum storage systems, two problems arise while repairing failures at the optimal repair bandwidth: there are the requirements of small field size q and the fragment M. If q and M are arbitrarily large, then the constructions are impractical due to the high computation complexity in decoding and the fast growing file size in storing.

[0053] Moreover, some conventional systems make the assumption that the same a and β are at each node. The assumption that a DSS should be homogeneous is very restrictive since in practical distributed storage systems, the storage nodes may be stored over the internet with different storage infrastructures and have routes of different capacities between them. The portion of a document delivered by one server should be proportional to its service rate. Thus, a slow server should deliver a small part of the document while a fast server should deliver a large part of the document. Freedom to download different amounts of data from different nodes helps to reduce the net download time and traffic congestion. Such systems will also be highly conducive for load-balancing across the nodes in the network. Therefore, another aspect of the present embodiment is to introduce non-homogeneity in the system and methods of the present embodiment, thereby expanding DSS construction to include non-homogeneous from the current framework of homogeneous distributed storage systems.

[0054] To minimize γ_1, let $d = n - 1$ and we get the lower bound for repair bandwidth γ_1 of single-node failure is shown in Equation 9:

$$\gamma_1 = \frac{n-1}{n-k}\frac{M}{k} \tag{9}$$

[0055] When the number of node failures equals $r \geq 2$, the optimal bound of repair bandwidth of (n,k) MSR codes is shown in Equation 10:

$$\gamma_r = \frac{r(d+r-1)}{(d+r-k)}\frac{M}{k} \tag{10}$$

[0056] Similarly, to minimize γ_r, let $d = n-r$, then the lower bound for repair bandwidth γ_r of r-node failure is shown in Equation 11:

$$\gamma_r = \frac{r(n-1)}{(n-k)}\frac{M}{k} \tag{11}$$

[0057] Note that a storage code where each node contains M/k worth of storage has an MDS property if a data collector (DC) can reconstruct the fragment M by connecting to any k out of n storage nodes.

Model of the non-homogeneous DSS

[0058] A non-homogeneous DSS with the parameter (n,k,h) is a distributed storage systems with h non-empty nodes based on (n,k) storing codes and the amount of data stored and downloaded from any nodes are variable. Node i in the network stores $\alpha_i \geq \frac{M}{k}$. When node i fails, the repair bandwidth of node i will be in Equation 12:

$$\gamma_1^{(i)} = \sum_{j \in \{n\} \setminus i} \beta_j \tag{12}$$

where β_j is downloading packets from node j.

[0059] In our model, we assume that there are more number of nodes, and each node has a lot of storage size, hence we consider the case $n \geq h$ and $\alpha_i \geq M/k$. When $n > h$, there are more redundant blocks than the storage nodes. The storage process has to decide which

node(s) to store more blocks. When $n = h, \alpha_i = \frac{M}{k}, \beta_j = \beta$ for all i, $j \neq i$, we obtain the traditional homogeneous DSS. It is clear that we must have $0 \leq \beta_j \leq \alpha_j$ for all $j \neq i$ since a node cannot transmit more information than it is storing. Different nodes may have different repair bandwidth and repair time.

[0060] Let $\mathbf{f}_1, \cdots, \mathbf{f}_k \in \mathbb{F}_q^N$ are k blocks divided from file of size M. After encoding, we receive $(n - k)$ parity blocks $\mathbf{p}_1, \cdots, \mathbf{p}_{n-k} \in \mathbb{F}_q^N$ where $\mathbf{p}_j = \mathbf{f}_1\mathbf{A}_{j1} + \mathbf{f}_2\mathbf{A}_{j2} + \cdots + \mathbf{f}_k\mathbf{A}_{jk}$. Here $\mathbf{A}_{ji}$ denote an $N \times N$ matrix of coding coefficients defined over finite field F_q for all $1 \leq i \leq k$ and $1 \leq j \leq n-k$. Let $x_i \geq 0$ denote the number of blocks of size N stored at storage node $i \in \{1, \cdots, n\}$, then the total amount of storage used over all nodes is n blocks in Equation 13:

$$\sum_{i=1}^{n} x_i = n \tag{13}$$

[0061] For any arbitrary (n,k) MDS codes, it can only correct maximum $(n - k)$ failure blocks. Therefore, the maximum blocks stored at each node must be no more than $(n- k)$. If we store beyond that, we will not be able to repair when a node fails as shown in Equation 14:

$$x_i = n - k \tag{14}$$

[0062] It should be noted that $x_i = 0$ means storage node i is an empty node. When $(x_1 = x_2 = \cdots = x_n = 1)$, it becomes the traditional data allocation in (n,k) MSR codes. Consider an example of $(n = 8, k = 5)$ MDS codes and a data storage system has 8 nodes with different bandwidth and storage capacity. Assume a file of size $M = 15$, then this file is divided into $k = 5$ blocks $\mathbf{f}_1, \cdots, \mathbf{f}_k$, each block containing $N = M/k = 3$ packets: $\mathbf{f}_i = [f_{i1}, \cdots, f_{iN}]^T$. Let $\mathbf{p}_j$ are parity blocks over finite field F_3 where $\mathbf{p}_j = \mathbf{f}_1\mathbf{A}_{j1} + \mathbf{f}_2\mathbf{A}_{j2} + \cdots + \mathbf{f}_k\mathbf{A}_{jk}$. Fig. 11 shows four different schemes of data allocation $(x_1, \cdots, x_n)$ named as traditional homogeneous, super-node non-homogeneous, partial-homogeneous, and minimum-spread

non-homogeneous. The data allocation of these four schemes corresponds to (1,1,1,1,1,1,1,1), (2,1,1,1,1,1,1,0), (2,2,2,2,0,0,0,0) and (3,2,3,0,0,0,0,0), respectively.

Data allocation for (n,k) MDS codes in non-homogeneous DSS

[0063] We motivate our investigation on data allocation $(x_1,x_2,\cdots,x_n)$ by considering in non-homogeneous DSS. In the following, we consider three different scenarios where a non-homogeneous DSS becomes more efficient than a homogeneous DSS.

[0064] Suppose that the download or recovery operation read y_i blocks from storage node i $(0 \leq y_i \leq x_i)$. We associate a weight w_i with node i, where w_i denotes the cost of downloading one block from node i and without loss of generality, let assume $w_1 \leq w_2 \leq \cdots \leq w_n$. Our objective is to seek an optimal allocation $(x_1,x_2,\cdots,x_n)$ that minimizes the download cost of k out of n blocks to reconstruct the original file as shown in Equation 15:

$$\begin{aligned} \text{minimize}_{y_i} \quad & C_{dc} = \sum_{i \in \{n\}} w_i y_i \\ \text{subject to :} \quad & \sum y_i \geq k \\ & y_i \leq x_i \leq n-k \end{aligned} \tag{15}$$

[0065] It can be seen that the download cost C_{dc} increases if we download much data from the high-cost nodes. Therefore, we have to store much data blocks in the low-cost nodes and less data on high-cost nodes. We remind that the possible of maximum blocks stored on each node is $(n-k)$ since we will not be able to repair when a node with more than $(n-k)$ blocks fails. It can be seen that we should allocate $(n-k)$ blocks on the first $\lfloor k/(n-k) \rfloor$ low-cost nodes to minimize C_{dc}. This will lead to the minimum-spread non-homogenous and partial-homogenous model in the next section.

[0066] Let $[p_1,\cdots,p_h]$ be the nodes' online probability of h nodes in the (n,k,h) DSS. Let the power set of h, 2^h, denote the set of all possible combinations of online nodes. Let $A \subset$

2^h represents one of these possible combinations. Then, we will use Q_A to represent the event that combination A occurs. Since node availabilities are independent, we have

$$Pr[Q_A] = \prod_{i \in A} p_i \prod_{j \in 2^h \setminus A} (1 - p_j) \tag{16}$$

[0067] Let $L_k \subset 2^h$ be the subset containing those combinations of available nodes which together store k different redundant blocks as shown in Equation 17:

$$L_k = \left\{ A : A \in 2^h, \sum_{i \in A} x_i \geq k \right\} \tag{17}$$

[0068] Since the retrieval process needs to download k different blocks out of the total n redundant blocks, the probability of successful recovery for an allocation $(x_1, x_2, \cdots, x_n)$ can be measured as Equation 18:

$$\begin{aligned} Pr[\text{successful recovery}] &= \sum_{A \in L_k} Pr[Q_A] \\ &= \sum_{A \in L_k} \left[\prod_{i \in A} p_i \prod_{j \in 2^h \setminus A} (1 - p_j) \right] \end{aligned} \tag{18}$$

[0069] The goal of optimal allocation $(x_1, x_2, \cdots, x_n)$ is to achieve the high data availability of original file in the non-homogeneous DSS. It is not hard to show that determining the recovery probability of a given allocation is computationally difficult (NP-hard). In one aspect, we consider some scenarios such as one node is super reliable and the others are the same reliable. This will lead to the super-node non-homogenous model proposed next.

[0070] After we decide $(x_1, x_2, \cdots, x_n)$ for allocation either minimize download cost or maximize availability, we should also consider optimal repair bandwidth for failure node. When node i fails, the repair bandwidth of node i will be Equation 19:

$$\gamma_1^{(i)} = \sum_{j \in \{n\} \setminus i} \beta_j \tag{19}$$

where β_j is downloading packets from node j.

[0071] In homogeneous DSS, one block is stored in one node. Hence single-node failure corresponds to single block lost. In non-homogeneous DSS, more than one blocks are stored in single node. Therefore, node i failure corresponds to multi-block x_i lost. Our objective is to seek an optimal allocation $(x_1, x_2, \cdots, x_n)$ that minimizes the repair bandwidth of node i as shown in Equation 20:

$$\begin{aligned} &\text{minimize}_{y_i} && \gamma_1^{(i)} \\ &\text{subject to :} && \gamma_1^{(i)} \leq \tfrac{(n-1)}{(n-k)} x_i \end{aligned} \tag{20}$$

[0072] The present embodiment presents a flexible framework of distributed storage systems named super-node non-homogeneous DSS. Super-node represents a storage node that has higher storage size, or higher communications bandwidth, or higher reliability than other nodes. In a practical system, the super-node may represent the local host, while other storage nodes are located remotely. Three schemes of super-node non-homogeneous DSS based on (k+2, k) MDS and non-MDS codes will be discussed hereinafter (i.e., Schemes A, B and C).

	Non-homogeneous		Homogeneous
	Proposed scheme A and B	Proposed scheme C	Traditional model
S. node			
s_1	$\mathrm{f}_1 \quad \mathrm{f}_2$	f_1	f_1
s_2	f_3	f_2	f_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_{k-1}	f_k	f_{k-1}	f_{k-1}
s_k	×	f_k	f_k
P. node			
p_1	$\mathrm{f}_1\mathrm{A}_1 + \cdots + \mathrm{f}_k\mathrm{A}_k$	$\mathrm{f}_1\mathrm{A}_1 + \cdots + \mathrm{f}_k\mathrm{A}_k$ $\mathrm{f}_1\mathrm{B}_1 + \cdots + \mathrm{f}_k\mathrm{B}_k$	$\mathrm{f}_1\mathrm{A}_1 + \cdots + \mathrm{f}_k\mathrm{A}_k$
p_2	$\mathrm{f}_1\mathrm{B}_1 + \cdots + \mathrm{f}_k\mathrm{B}_k$	×	$\mathrm{f}_1\mathrm{B}_1 + \cdots + \mathrm{f}_k\mathrm{B}_k$

Table 9

Table 9 sets out a comparison of the three schemes of super-node non-homogeneous model versus a traditional homogeneous model based on (k+2, k) MDS codes where S and P are the abbreviations for systematic and parity, respectively. Here, $\mathrm{f}_i \in \mathbb{F}_q^{1\times N}$ and

$\mathbf{A}_i, \mathbf{B}_i \in \mathbb{F}_q^{N \times N}$, for all $1 = i = k$. Note that all of super-node schemes A, B and C use only k+1 storage nodes to store k+2 packets. Note also that schemes A and B both store two systematic data $\mathbf{f}_1$ and $\mathbf{f}_2$ at the same storage node s_1, while scheme C stores two parity data at the same storage node. A similar idea can be extended to k+1 or k+2 storage nodes to store k+3 packets, or any further extension. Systems and methods in accordance with these aspects of the present embodiment achieve optimal repair bandwidth $\frac{k+1}{2}\frac{M}{k}$ at smaller finite field q and 4 times smaller file size M than traditional homogeneous systems. In addition, the relax MDS property of (k+2, k) storage codes allows achievement of smaller repair bandwidths for one failure at $\frac{M}{2} < \frac{k+1}{2}\frac{M}{k}$.

Super-node Scheme A: Store Two Systematic Data at the Same Storage Node (MDS Code).

[0073] In discussing a repair of a one-node failure (the case of big node s_1 failing is considered as a two-node failure which will be discussed in more detail later), it is assumed that node s_2 that contains $\mathbf{f}_3$ is failed. For simplicity, the case (n = 5, k = 3) is initially considered. To recover desired data $\mathbf{f}_3$, the following equations (see Equation 21) are downloaded from two survival parity nodes, where the $\mathbf{V}^1$, $\mathbf{V}^2$ matrices are based on the failure node. To repair a different node, different $\mathbf{V}^1$, $\mathbf{V}^2$ are needed which can be pre-calculated and stored in a controller as shown in Equation 21:

$$\begin{matrix} \mathbf{f}_1\mathbf{A}_1\mathbf{V}^1 + \mathbf{f}_2\mathbf{A}_2\mathbf{V}^1 + \mathbf{f}_3\mathbf{A}_3\mathbf{V}^1 \\ \mathbf{f}_1\mathbf{B}_1\mathbf{V}^2 + \mathbf{f}_2\mathbf{B}_2\mathbf{V}^2 + \mathbf{f}_3\mathbf{B}_3\mathbf{V}^2 \end{matrix} \qquad (21)$$

where $\mathbf{A}_i, \mathbf{B}_i \in \mathbb{F}_q^{N \times N}$ for all $1 = i = k$ and $\mathbf{V}^1, \mathbf{V}^2 \in \mathbb{F}_q^{N \times \frac{N}{2}}$. It can be seen that the term ($\mathbf{f}_1\mathbf{A}_1\mathbf{V}^1$ + $\mathbf{f}_2\mathbf{A}_2\mathbf{V}^1$) and ($\mathbf{f}_1\mathbf{B}_1\mathbf{V}^2$ + $\mathbf{f}_2\mathbf{B}_2\mathbf{V}^2$) are removable by downloading (N/2 + N/2) packets from big node 1 (See FIG. 7). Therefore, the desired data $\mathbf{f}_3$ can be recovered if the following rank constraint is satisfied in Equation 22:

$$\text{rank}\,[\mathbf{A}_3\mathbf{V}^1,\ \mathbf{B}_3\mathbf{V}^2] = N \qquad (22)$$

[0074] For general (k+2, k) case, the optimal repair bandwidth of 1 failure will be $\frac{k+1}{2}\frac{M}{k}$. To recover the desired data $\mathbf{f}_3$ we have to use Equation 23:

$$\begin{matrix}\mathbf{f}_1\mathbf{A}_1\mathbf{V}^1 + \mathbf{f}_2\mathbf{A}_2\mathbf{V}^1 + \mathbf{f}_3\mathbf{A}_3\mathbf{V}^1 + \cdots + \mathbf{f}_k\mathbf{A}_k\mathbf{V}^1 \\ \mathbf{f}_1\mathbf{B}_1\mathbf{V}^2 + \mathbf{f}_2\mathbf{B}_2\mathbf{V}^2 + \mathbf{f}_3\mathbf{B}_3\mathbf{V}^2 + \cdots + \mathbf{f}_k\mathbf{B}_k\mathbf{V}^2\end{matrix} \quad (23)$$

Similarly, the terms ($\mathbf{f}_1\mathbf{A}_1\mathbf{V}^1$ + $\mathbf{f}_2\mathbf{A}_2\mathbf{V}^1$) and ($\mathbf{f}_1\mathbf{B}_1\mathbf{V}^2$ + $\mathbf{f}_2\mathbf{B}_2\mathbf{V}^2$) are removable by downloading (N/2 + N/2) packets from big node 1. The following condition must be satisfied to achieve the optimal repair bandwidth in Equation 24:

$$\begin{matrix}\text{rank}\,[\mathbf{A}_3\mathbf{V}^1,\ \mathbf{B}_3\mathbf{V}^2] = N \\ \text{rank}\,[\mathbf{A}_4\mathbf{V}^1,\ \mathbf{B}_4\mathbf{V}^2] = \frac{N}{2} \\ \vdots \\ \text{rank}\,[\mathbf{A}_k\mathbf{V}^1,\ \mathbf{B}_k\mathbf{V}^2] = \frac{N}{2}\end{matrix} \quad (24)$$

[0075] To relax the complexity of the constraints found in Equation 24, we set $\mathbf{A}_i = \mathbf{I}_N$ and $\mathbf{V}^1 = \mathbf{V}^2$, then obtain Equation 25:

$$\begin{matrix}\text{rank}\,[\mathbf{B}_3\mathbf{V}^1,\ \mathbf{V}^1] = N \\ \text{rank}\,[\mathbf{B}_4\mathbf{V}^1,\ \mathbf{V}^1] = \frac{N}{2} \\ \vdots \\ \text{rank}\,[\mathbf{B}_k\mathbf{V}^1,\ \mathbf{V}^1] = \frac{N}{2}\end{matrix} \quad (25)$$

The problem of finding matrix $\mathbf{B}_i$ is a problem similar to typical homogeneous DSS. However, in accordance with the present embodiment, only (k - 2) equations need to be solved. Therefore, the fragment size and finite field will be smaller $M = 2^{k-1}k$, and $q = 2k-1$. This means that the fragment size is reduced to one-fourth of the traditional homogeneous model. This advantageously allows reduction in the minimum size unit of the storing file and the complexity of computation in the smaller finite field.

[0076] In the case where the first parity node p_1 fails, a change of variables is made to obtain a new representation for the code in accordance with the present embodiment such that the first parity p_1 becomes a systematic node in the new representation. The change of variables is made as set out in Equation 26:

$$\sum_{i=1}^{k} f_i = y_3,\ f_n = y_s \quad \text{for } 1 \le s \ne 3 \le k \tag{26}$$

And Equation 26 is solved by replacing f_3 in terms of the y_i variables and obtaining Equation 27:

$$f_3 = y_3 - \sum_{s=1, s\neq 3}^{k} y_s \tag{27}$$

[0077] The problem of repairing a first parity is equivalent to repairing a systematic node y_3 in the new presentation. Note that y_1, y_2 are stored in the same node since they correspond to f_1, f_2. To repair y_3, download is made in accordance with Equation 28:

$$\begin{array}{l}(-y_1) + (-y_2) + y_3 + \cdots + (-y_k) \\ (B_1 - B_3)y_1 + (B_1 - B_3)y_2 + B_3 y_3 + \cdots + (B_k - B_3)y_k\end{array} \tag{28}$$

Again, the V^1, V^2 matrices need to satisfy the conditions of Equation 28 in order to achieve the optimal repair bandwidth in Equation 29:

$$\begin{array}{l}\text{rank}\,[B_3 V^1,\ V^1] = N \\ \text{rank}\,[(B_4 - B_3)V^1,\ V^1] = \frac{N}{2} \\ \vdots \\ \text{rank}\,[(B_k - B_3)V^1,\ V^1] = \frac{N}{2}\end{array} \tag{29}$$

[0078] In the same manner, the code in accordance with the present embodiment is rewritten in a form where the second parity is a systematic node in some presentation, as shown in Equation 30:

$$\begin{bmatrix} I_N & 0 & 0 & \cdots & 0 \\ 0 & I_N & 0 & \cdots & 0 \\ 0 & 0 & I_N & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & I_N \\ I_N & I_N & I_N & \cdots & I_N \\ B_1 & B_2 & B_3 & \cdots & B_k \end{bmatrix} \mathbf{f} = \begin{bmatrix} I_N & 0 & 0 & \cdots & 0 \\ 0 & I_N & 0 & \cdots & 0 \\ 0 & 0 & I_N & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & I_N \\ B_1' & B_2' & B_3' & \cdots & B_k' \\ I_N & I_N & I_N & \cdots & I_N \end{bmatrix} \mathbf{f}' \tag{30}$$

where f' is a full rank row transformation of f. The repair solution is determined in the same manner as handled above in regards to the first parity repair to achieve the optimal repair bandwidth for the second parity of the code.

Super-node Scheme B: Store Two Systematic Data at the Same Storage Node (Non-MDS Code).

[0079] Scheme B uses the same model as scheme A. However, we can achieve the repair bandwidth for 1 failure below the optimal bound in this non-homogeneous model if the term ($\mathbf{f}_1\mathbf{A}_1\mathbf{V}^1 + \mathbf{f}_2\mathbf{A}_2\mathbf{V}^1$) and ($\mathbf{f}_1\mathbf{B}_1\mathbf{V}^2 + \mathbf{f}_2\mathbf{B}_2\mathbf{V}^2$) are the same or if the following constraints are satisfied: $\mathbf{A}_1\mathbf{V}^1 = \mathbf{B}_1\mathbf{V}^2$, $\mathbf{A}_2\mathbf{V}^1 = \mathbf{B}_2\mathbf{V}^2$. The following example is used to present the idea of repairing 1 failure below the optimal bandwidth bound for the simple case k=3, n=5. Consider $\mathbf{f}_1 = [a_1, a_2]^T, \mathbf{f}_2 = [b_1, b_2]^T, \mathbf{f}_3 = [c_1, c_2]^T$ and $\mathbf{p}_1 = \mathbf{f}_1\mathbf{A}_1 + \mathbf{f}_2\mathbf{A}_2 + \mathbf{f}_3\mathbf{A}_3$, $\mathbf{p}_2 = \mathbf{f}_1\mathbf{B}_1 + \mathbf{f}_2\mathbf{B}_2 + \mathbf{f}_3\mathbf{B}_3$ are the systematic and parity data of a (5,3) storage code over a finite field $\mathbb{F}_3$ in Equation 31:

$$\mathrm{A}_1 = \begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix}, \mathrm{A}_2 = \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix}, \mathrm{A}_3 = \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix}, \mathrm{B}_1 = \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix}, \mathrm{B}_2 = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}, \mathrm{B}_3 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \tag{31}$$

It can be seen that any single failure (systematic or parity node) except the big node can be repaired with a bandwidth below the optimal bound $\frac{k+1}{2}\frac{M}{k}$. FIG. 8 shows the process of using 2 projection vectors in Equation 32:

$$\mathrm{V}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathrm{V}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{32}$$

for repairing 1 systematic failure below the optimal bandwidth bound. It is straightforward for the general case (k + 2, k). Therefore, the solution for scheme B can be found in a manner similar to scheme A. However, in scheme B the MDS property of the storage code is not kept since reconstruction of the original information from existing nodes cannot be made if the big node or 2 small nodes fail.

Super-node Scheme C: Store Two Parity Data at the Same Storage Node (MDS Code).

[0080] Similar to scheme A, we first consider the case where n=5, k=3 for simplicity. Without loss of generality, assume that node 1 is failed and 2 parity packets $\mathbf{p}_1$, $\mathbf{p}_2$ are stored at the same parity node. To recover $\mathbf{f}_1$, Equation 33 is followed after eliminating $\mathbf{f}_2$ and $\mathbf{f}_3$ from the parity node:

$$\begin{cases} \mathbf{f}_1\mathbf{A}_1\mathbf{V}^1 + \mathbf{f}_2\mathbf{A}_2\mathbf{V}^1 + \mathbf{f}_3\mathbf{A}_3\mathbf{V}^1 \\ \mathbf{f}_1\mathbf{B}_1\mathbf{V}^2 + \mathbf{f}_2\mathbf{B}_2\mathbf{V}^2 + \mathbf{f}_3\mathbf{B}_3\mathbf{V}^2 \end{cases} \Longrightarrow \begin{cases} \mathbf{f}_1\mathbf{C}_1\mathbf{V}^1 + \mathbf{f}_2\mathbf{C}_2\mathbf{V}^1 \\ \mathbf{f}_1\mathbf{D}_1\mathbf{V}^2 + \mathbf{f}_3\mathbf{D}_2\mathbf{V}^2 \end{cases} \tag{33}$$

where $\mathbf{C}_i, \mathbf{D}_i \in \mathbb{F}_q^{N\times N}$ for i = 1, 2 and $\mathbf{C}_1 = \mathbf{A}_1\mathbf{A}_3^{-1} - \mathbf{B}_1\mathbf{B}_3^{-1}$, $\mathbf{C}_2 = \mathbf{A}_2\mathbf{A}_3^{-1} - \mathbf{B}_2\mathbf{B}_3^{-1}$, $\mathbf{D}_1 = \mathbf{A}_1\mathbf{A}_2^{-1} - \mathbf{B}_1\mathbf{B}_2^{-1}$, and $\mathbf{D}_2 = \mathbf{A}_3\mathbf{A}_2^{-1} - \mathbf{B}_3\mathbf{B}_2^{-1}$. It can be seen that the term $\mathbf{f}_2\mathbf{C}_2\mathbf{V}^1$ and $\mathbf{f}_3\mathbf{D}_2\mathbf{V}^2$ are removable by downloading (N/2 + N/2) packets from the parity node (See FIG. 9). Therefore, the desired data $\mathbf{f}_1$ can be recovered if the following rank constraint is satisfied in Equation 34:

$$\operatorname{rank}[\mathbf{C}_1\mathbf{V}^1,\ \mathbf{D}_1\mathbf{V}^2] = N \tag{34}$$

[0081] For a general (k+2, k) case, we set $\mathbf{A}_i = \mathbf{I}_N$ for all $i = N$ (similar to scheme A). To recover the desired data $\mathbf{f}_1$, the Equation 35 is reduced from the parity node:

$$\begin{matrix} \mathbf{f}_1(\mathbf{B}_1 - \mathbf{B}_3) + \mathbf{f}_2(\mathbf{B}_2 - \mathbf{B}_3) + \mathbf{f}_4(\mathbf{B}_4 - \mathbf{B}_3) + \cdots + \mathbf{f}_k(\mathbf{B}_k - \mathbf{B}_3) \\ \mathbf{f}_1(\mathbf{B}_1 - \mathbf{B}_2) + \mathbf{f}_3(\mathbf{B}_3 - \mathbf{B}_2) + \mathbf{f}_4(\mathbf{B}_4 - \mathbf{B}_2) + \cdots + \mathbf{f}_k(\mathbf{B}_k - \mathbf{B}_2) \end{matrix} \tag{35}$$

[0082] The condition of Equation 36 must be satisfied to achieve the optimal repair bandwidth $\frac{k+1}{2}\frac{M}{k}$:

$$\begin{matrix} \operatorname{rank}[(\mathbf{B}_1 - \mathbf{B}_2)\mathbf{V}^1,\ (\mathbf{B}_1 - \mathbf{B}_3)\mathbf{V}^2] = N \\ \operatorname{rank}[(\mathbf{B}_4 - \mathbf{B}_2)\mathbf{V}^1,\ (\mathbf{B}_4 - \mathbf{B}_3)\mathbf{V}^2] = \frac{N}{2} \\ \vdots \\ \operatorname{rank}[(\mathbf{B}_k - \mathbf{B}_2)\mathbf{V}^1,\ (\mathbf{B}_k - \mathbf{B}_3)\mathbf{V}^2] = \frac{N}{2} \end{matrix} \tag{36}$$

Repair 2 failures for Super-node Schemes A and C

[0083] It is trivial to repair the big node s_1 at the repair bandwidth of M by fully downloading data from survival nodes. To repair two small node failures at the optimal

repair bandwidth, one solution is shown in FIGs. 10A and 10B. For scheme A, download the k packet from the survival nodes; then the original file can be recovered due to the properties of MDS codes. Therefore, the data of nodes s_2 and p_1 can be obtained and stored in new node s_2. Next, the data of failure node p_1 is forwarded to a newcomer node. The total repair bandwidth will be $\gamma_2 = M + M/k$. The optimal repair bandwidth for 2 failure nodes for scheme C can be achieved in the same manner. It should be noted that the failure of 1 big node and 1 small node cannot be repaired since it can be regarded as 3 single-node failure nodes, which is beyond the correcting ability of $(k + 2, k)$ MDS codes.

	Scheme A&C	Scheme B	Alex [1]	Perm. code [5]	Tamo [13]	C.R.C [9]
	$M = 2^{k-1}k$ $q \geq 2k-1$	$M = 2^{k-1}k$ $q \geq 2k-1$	$M = 2^{k+1}k$ $q \geq 2k+3$	$M = 2^k k$ $q \geq 2k+1$	$M = 2^k k$ $q \geq 2k+1$	$M = 2k$ $q \geq n$
1 failure	$\gamma = \frac{M}{k}\frac{k+1}{2}$	$\gamma = \frac{M}{2}$	$\gamma = \frac{M}{k}\frac{k+1}{2}$	$\gamma = \frac{M}{k}\frac{k+1}{2}$	$\gamma = \frac{M}{k}\frac{k+1}{2}$	N.A
2 failures	$\gamma = M + \frac{M}{k}$	N.A	$\gamma = M + \frac{M}{k}$	$\gamma = M + \frac{M}{k}$	$\gamma = M + \frac{M}{k}$	$\gamma = M + \frac{M}{k}$

Table 10

[0084] Thus it can be seen that optimal repair bandwidths for 1 failure can be achieved at a smaller finite field q and at a four times smaller fragment size M than conventional schemes. In addition, repairing 1 failure using non-MDS codes can achieve $\frac{M}{2k}$ smaller bandwidth than the optimal bound. A summary is presented in Table 10 for the present schemes and various conventional technologies.

[0085] Using the example $n = 5$, $k = 3$, assume a data file, denoted as FILE, of size 48 GB is needed to store across the distributed storage system. In accordance with schemes A and C, the file is divided into four fragments of size $M_1 = 12$. These fragments are stored across k + 1 = 4 nodes in the non-homogeneous DSS. If a small node fails, the repair bandwidth will be $4 \times \frac{M_1}{k}\frac{k+1}{2} = 32\text{GB}$. If two failure nodes occur, the repair bandwidth will be $4 \times \left(M_1 + \frac{M_1}{k}\right) = 64\text{GB}$. To update one fragment M_1 of the file, the update

bandwidth will be $\frac{M_1}{k}n$ = 20GB. The same results are achieved for both schemes A and C in repairing failures and updating information. In regards to scheme B, the file is divided into four fragments of size M_1 = 12 (similar to the division in regards to schemes A and C). If a small node fails, the repair bandwidth will be $4 \times \frac{M_1}{2}$ = 24GB. To update one fragment M_1 of the file, the update bandwidth will be $\frac{M_1}{k}n$ = 20GB.

	Scheme A&C	Scheme B	Alex[1]	Perm. code[5]	Tamo [13]	C.R.C [9]
	$M_1 = 12$ $q \geq 5$	$M_1 = 12$ $q \geq 5$	$M_2 = 48$ $q \geq 9$	$M_3 = 24$ $q \geq 7$	$M_4 = 24$ $q \geq 7$	$M_5 = 6$ $q \geq 5$
1 failure	$\gamma = 32$	$\gamma = 24$	$\gamma = 32$	$\gamma = 32$	$\gamma = 32$	N.A
2 failures	$\gamma = 64$	N.A	$\gamma = 64$	$\gamma = 64$	$\gamma = 64$	$\gamma = 64$
update $\leq$ 12GB data	$\delta = 20$	$\delta = 20$	$\delta = 80$	$\delta = 40$	$\delta = 40$	$\delta = 10$

Table 11

[0086] In regards to a first conventional system (denoted in Tables 10 and 11 as Alex), the file is divided into one fragment of size M_2 = 48 such as FILE = M_2. The repair bandwidth of one failure node for this systems is, therefore, $1 \times \left(\frac{M_2}{k}\frac{k+1}{2}\right)$ = 32GB and for two failure nodes $1 \times \left(M_2 + \frac{M_2}{k}\right)$ = 64GB is required. The update bandwidth for one fragment M_2 will be $\frac{M_2}{k}n$ = 80GB. The repair and update bandwidth for other conventional methods are computed in a similar manner and shown in Table 11. Note that the C.R.C. method cannot repair one failure with optimal bandwidth. All of the methods require similar bandwidth for repairing failures except scheme B. Moreover, schemes A, B and C have advantages in updating small parts of the file as compared with the conventional methods (except the C.R.C. method). The C.R.C. method, however, is not practical since it cannot achieve the optimal repair bandwidth in the case of 1 failure node. Permutation method and MDS array method are also impractical since they can only repair the systematic nodes.

[0087] Thus, the non-homogeneous DSS in accordance with the present embodiment provides a flexible framework for distributed storage systems. Two schemes of storing data using a $(k + 2, k)$ MDS codes can achieve optimal repair bandwidth $(k+1)/2 \cdot M/k$ at smaller finite field q and a four times smaller fragment M than prior art systems. The smaller M and q also help the non-homogeneous DSS to save update bandwidth more efficiently than traditional methods. Moreover, scheme B can achieve one failure repair bandwidth at $\frac{M}{2k}$ smaller than the optimal bandwidth bound.

Numerical case study

[0088] To compare the data availability, we examine a scenario of node online probability where the online probability of super-node is greater than the other node $p_1 \geq p_2 = p_3 = \cdots = p_n = p$.

[0089] The data availability of homogeneous Pr_{homo} and non-homogeneous $Pr_{non\text{-}homo}$ DSS can be computed by Equation 37 and 38:

$$Pr_{homo} = p^{k+1} + (k+1)(1-p)p^k + \frac{k(k+1)}{2} p_1 (1-p)^2 p^{k-1} \quad (37)$$

$$Pr_{non-homo} = p^k + kp_1(1-p)p^{k-1} + \frac{k(k-1)}{2} p_1 (1-p)^2 p^{k-2} \quad (38)$$

[0090] Let $p_1 = \chi p$ where $\chi \geq 1$. The condition $Pr_{non\text{-}homo} \geq Pr_{homo}$ will induce $\chi \geq p/(p+\frac{1}{2}(1-p)[(k-1)-(k+1)p])$. It can be seen that if $p \leq \frac{k-1}{k+1}$, then $p/(p + \frac{1}{2}(1 - p)[(k - 1) - (k + 1)p]) \leq 1 \leq \chi$. Therefore, $Pr_{non\text{-}homo} \geq Pr_{homo}$ for all $p \leq \frac{k-1}{k+1}$. We run the simulations for the case of $k = 4$, $p = 0.6$ and $p = 0.65$ and obtain the result in Fig. 12. It can be seen that for $p = \frac{k-1}{k+1} = 0.6$, data availability of non-homogeneous DSS scheme outperforms the homogeneous DSS scheme. For $p = 0.65 > \frac{k-1}{k+1}$, the non-homogeneous schemes also have a big improvement when p_1 has a high online availability. Therefore, it can be seen that our proposed non-homogeneous DSS schemes achieve a higher data

availability than the traditional homogeneous DSS. The gap between the two becomes larger when the online availability of the super node increases, e.g. when p_1 is greater than 25% of p, the data availability of the proposed non-homogenous over homogenous DSS is increased by 10%.

Partial-Homogeneous DSS

[0091] In this scheme, all nonzero nodes store the same blocks of information. The data allocation $(x_1,x_2,\cdots,x_n)$ of this scheme corresponds to Equation 39:

$$\begin{cases} x_1 = x_2 = \cdots = x_h = \dfrac{n}{h} \\ \text{while } 1 \le \dfrac{n}{h} \le n-k \end{cases} \tag{39}$$

[0092] In traditional homogeneous, the intuitive approach of spreading n blocks maximally over n nodes, i.e., assigning $x_i = 1$ for all $1 \le i \le n$, turns out to be optimal. In the non-homogeneous, the optimal allocation may not be maximal spread. For maximum reliability and minimum the download cost, we would therefore need to find an optimal allocation x_1. It should be note that this scheme corresponds to the case of storage budget equal to the file size. However, our partial-homogeneous scheme considers how to achieve the optimal repair bandwidth when a node fails. This leads to the requirement of x_i strict to integer number for all $1 \le i \le n$. In order to download cost of original file, we have to download full from low-cost node and total download blocks equal k. The corresponding download cost of original file will be in Equation 40:

$$C_{dc} = \sum_{1 \le i \le h-1} \frac{w_i n}{h} \tag{40}$$

[0093] In special case $n = h(n - k)$ or $x_1 = n - k$, any single node failure is equivalent to losing $(n-k)$ blocks. Since $k = (h - 1)(n - k)$, the new incoming node have to collects k

blocks from $h - 1$ nodes to repair any single-node failure $\gamma_1^{(i)} = \sum_{1 \le j \ne i \le h} (n-k) = k$. The corresponding download cost of the original file will be in Equation 41

$$C_{dc} = \sum_{1 \le i \le h-1} w_i(n-k) \tag{41}$$

[0094] Fig. 13 shows an example of (n = 6, k = 4, h = 3) non-homogeneous DSS using the same (6,4) MDS code in the previous example. In this case, h = 3 and we use only 3 nodes to store data (x_1 = x_2 = x_3 = 2). To repair the third node failure, we have to download k = 4 blocks f_1,f_2,f_3,f_4 from node 1 and node 2. The download cost will be C_{dc} = $2w_1$ + $2w_2$. It can be seen that repairing failure node in this case is similar to the traditional homogeneous DSS.

Minimum-Spread Non-Homogeneous DSS

[0095] In this scheme, we try to store much possible data on each node. It is named as minimum-spread non-homogeneous since we use minimum number of active nodes than any other schemes. Assume n = $(h-1)(n-k)+r$ where $1 \le r < n-k$, we have two kind of nodes which store $(n - k)$ blocks and r blocks. Due to information blocks is more important than redundancy blocks, it is recommended that k information blocks should spread on lowest cost nodes and $(n-k)$ parity blocks should stay on the same node with the highest cost. The data allocation ($x_1,x_2,\cdots,x_n$) of this scheme corresponds to Equation 42:

$$\begin{cases} x_1 = x_2 = \cdots = x_{h-2} = n-k \\ x_{h-1} = r, \quad x_h = n-k \end{cases} \tag{42}$$

[0096] The download cost of original file is optimal and is shown in Equation 43.

$$C_{dc} = \sum_{i=1}^{h-2} w_i(n-k) + w_{h-1}r \tag{43}$$

[0097] When a node with $(n-k)$ blocks fails, we have to download k blocks from survival nodes. In the case of repairing failure node of r blocks, interference alignment has been applied to achieve the optimal repair bandwidth of MSR codes by aligning the various interferences independently. Here, we show one possible solution of repair nodes with r blocks by using interference alignment in non-homogeneous DSS.

1. $h = 2 \rightarrow r = k$, then $k \leq \frac{n}{2}$ and $k \leq n - k$ (low code rate). The optimal data allocation will be $(x_1 = n - k, x_2 = k)$. It is trivial to repair any single node, we only have to download k blocks from the survival node.

2. $h \geq 3$ and $0 < r < n - k$, *then* $\frac{n}{2} < k$ and $(n - k) \leq k$ (high code-rate). When node i fails, the optimal repairing process is as follow:

   - $1 \leq i \leq h - 2$ and $i = h$, then $y_j = x_j$ for $i \neq j$, the total of download blocks for repairing node i is k blocks as shown in Equation 44

$$\sum_{1 \leq j \neq i \leq h} y_j = (h-1)(n-k) = k \tag{44}$$

   - $i = h - 1$, the total of download blocks is depend on the minimum of k and the bound $\frac{r(n-1)}{n-k}$. The condition $\frac{r(n-1)}{n-k} > k$ corresponds to $k^2 - nk + r(n - 1) > 0$ or $k > \frac{h-1}{h}n$. Therefore, when $k > \frac{h-1}{h}n$, we need to download k blocks to repair $(h - 1)$-th node as shown in Equation 45:

$$y_j = \begin{cases} x_j, & \text{if } 1 \leq j \leq h-2 \\ r, & \text{if } j = h-1 \\ 0, & \text{otherwise} \end{cases} \tag{45}$$

[0098] When $\frac{n}{2} < k \leq \frac{h-1}{h}n$, to repair node $(h - 1)$ we need to download $(h - 1)r$ blocks as shown in Equation 46:

$$y_j = \begin{cases} r, & \text{if } 1 \le j \le h-2 \text{ and } j = h \\ 0, & \text{otherwise} \end{cases} \tag{46}$$

[0099] Table 13 summarize the repair bandwidth for any failure node in the minimum-spread model.

SUMMARY OF REPAIR BANDWITH FOR ANY FAILURE NODE IN THE MINIMUM SPREAD NON-HOMOGENEOUS DSS BASED ON (n,k) MSR CODES $n=(h-1)(n-k)+r$.

	Node	Number of blocks	Min. Spread	Traditional MSR code
$h=2, k \le \frac{n}{2}$	1	$n-k$	k	k
	2	k	k	k
$h \ge 3, \frac{n}{2} < k \le \frac{h-1}{h}n$	$1,\cdots,(h-2)$	$n-k$	k	k
	$(h-1)$	r	$(h-1)r$	$(h-1)r+\frac{r(r-1)}{n-k}=\frac{r(n-1)}{n-k}$
	h	$n-k$	k	k
$h \ge 3, k > \frac{h-1}{h}n$	$1,\cdots,(h-2)$	$n-k$	k	k
	$(h-1)$	r	k	k
	h	$n-k$	k	k

Table 13

Minimum-spread scheme for (k + 3,k) MSR codes

[00100] First, consider the case $k = 5$. Assume a file of size M is divided into $k = 5$ blocks $\mathbf{f}_1,\cdots,\mathbf{f}_k$, each block containing $N = \frac{M}{k}$ packets: $\mathbf{f}_i = [f_{i1}, \cdots, f_{iN}]^T$. After encoding them into k+3 encoded blocks, we store them across the distributed storage as below.

- The first node stores 3 systematic blocks $\mathbf{f}_1,\mathbf{f}_2,\mathbf{f}_3$
- The second node stores 2 systematic blocks $\mathbf{f}_4,\mathbf{f}_5$
- The parity node stores 3 redundancy blocks $\mathbf{p}_1 = \mathbf{f}_1\mathbf{A}_{11} + \cdots + \mathbf{f}_k\mathbf{A}_{1k}$, $\mathbf{p}_2 = \mathbf{f}_1\mathbf{A}_{21} + \cdots + \mathbf{f}_k\mathbf{A}_{2k}$ and $\mathbf{p}_3 = \mathbf{f}_1\mathbf{A}_{31} + \cdots + \mathbf{f}_k\mathbf{A}_{3k}$.

[00101] Here, $\mathbf{A}_{1i}$, $\mathbf{A}_{2i}$ and $\mathbf{A}_{3i} \in F_q^{N\times N}$ for all $1 \le i \le k$. To recover the data block $\mathbf{f}_1,\mathbf{f}_2$ in the second node, we have to download the Equations 47 and 48 from the last node:

$$\begin{cases} \mathbf{f}_1\mathbf{A}_{11}\mathbf{V}^1 + \mathbf{f}_2\mathbf{A}_{12}\mathbf{V}^1 + \cdots + \mathbf{f}_5\mathbf{A}_{15}\mathbf{V}^1 \\ \mathbf{f}_1\mathbf{A}_{21}\mathbf{V}^2 + \mathbf{f}_2\mathbf{A}_{22}\mathbf{V}^2 + \cdots + \mathbf{f}_5\mathbf{A}_{25}\mathbf{V}^2 \\ \mathbf{f}_1\mathbf{A}_{31}\mathbf{V}^3 + \mathbf{f}_2\mathbf{A}_{32}\mathbf{V}^3 + \cdots + \mathbf{f}_5\mathbf{A}_{35}\mathbf{V}^3 \end{cases} \quad (47)$$

$$\begin{cases} \mathbf{f}_1\mathbf{A}_{11}\mathbf{W}^1 + \mathbf{f}_2\mathbf{A}_{12}\mathbf{W}^1 + \cdots + \mathbf{f}_5\mathbf{A}_{15}\mathbf{W}^1 \\ \mathbf{f}_1\mathbf{A}_{21}\mathbf{W}^2 + \mathbf{f}_2\mathbf{A}_{22}\mathbf{W}^2 + \cdots + \mathbf{f}_5\mathbf{A}_{25}\mathbf{W}^2 \\ \mathbf{f}_1\mathbf{A}_{31}\mathbf{W}^3 + \mathbf{f}_2\mathbf{A}_{32}\mathbf{W}^3 + \cdots + \mathbf{f}_5\mathbf{A}_{35}\mathbf{W}^3 \end{cases} \quad (48)$$

[00102] where $\mathbf{A}_{ji} \in \mathbb{F}_q^{N\times N}$ and $\mathbf{V}^j, \mathbf{W}^j \in \mathbb{F}_q^{N\times\frac{N}{n-k}}$ for all $1 \le i \le k$ and $1 \le j \le n-k$. It can be seen from Fig. 14 that the term $(\mathbf{f}_1\mathbf{A}_{i1}\mathbf{V}^i + \mathbf{f}_2\mathbf{A}_{i2}\mathbf{V}^i + \mathbf{f}_3\mathbf{A}_{i3}\mathbf{V}^i)$ and $(\mathbf{f}_1\mathbf{A}_{i1}\mathbf{W}^i + \mathbf{f}_2\mathbf{A}_{i2}\mathbf{W}^i + \mathbf{f}_3\mathbf{A}_{i3}\mathbf{W}^i)$ are removable by downloading $\frac{N}{n-k} = 1$ packets from second node. Therefore, the desired data $\mathbf{f}_4$ and $\mathbf{f}_5$ can be recovered by solving Equation 49:

$$\begin{cases} \mathbf{f}_4\mathbf{A}_{14}\mathbf{V}^1 + \mathbf{f}_5\mathbf{A}_{15}\mathbf{V}^1 \\ \mathbf{f}_4\mathbf{A}_{24}\mathbf{V}^2 + \mathbf{f}_5\mathbf{A}_{25}\mathbf{V}^2 \\ \mathbf{f}_4\mathbf{A}_{34}\mathbf{V}^3 + \mathbf{f}_5\mathbf{A}_{35}\mathbf{V}^3 \\ \mathbf{f}_4\mathbf{A}_{14}\mathbf{W}^1 + \mathbf{f}_5\mathbf{A}_{15}\mathbf{W}^1 \\ \mathbf{f}_4\mathbf{A}_{24}\mathbf{W}^2 + \mathbf{f}_5\mathbf{A}_{25}\mathbf{W}^2 \\ \mathbf{f}_4\mathbf{A}_{34}\mathbf{W}^3 + \mathbf{f}_5\mathbf{A}_{35}\mathbf{W}^3 \end{cases} \quad (49)$$

[00103] It can be seen that the number of unknown variables and the number of equations in (49) are $2N$. Therefore, we can recover the desired data $\mathbf{f}_4$ and $\mathbf{f}_5$ by solving the Equation 49. The repair bandwidth process requires $4N$. The repair bandwidth of node 3 will be equivalent to 2 failure nodes in the traditional homogeneous and the bound of repair bandwidth for 2 failures will be $\gamma = 2 \times \frac{(n-1)}{(n-k)}\frac{M}{k} = 4N + \frac{2}{3}N$. It can be seen that the minimum-spread non-homogeneous save $\frac{2}{n-k}\frac{M}{k}$ bandwidth than traditional homogeneous DSS.

[00104] In the general $(n = k + 3, k)$ MSR codes, the repair bandwidth of $(n-k)$-block node is k blocks while the repair bandwidth of r-block node in minimum-spread non-

homogeneous is reduced by $\frac{r(r-1)}{n-k}\frac{M}{k}$ than the traditional homogeneous DSS by applying the same technique as seen in the case of $k = 5$. It can be concluded that minimum-spread non-homogeneous DSS can repair any single node with the optimal repair bandwidth.

Performance analysis for minimum-spread non-homogeneous DSS

[00105] As compared with the traditional MSR codes, our scheme can achieve better download cost of original file and smaller repair bandwidth of r-blocks node. Our scheme can reduce $\frac{r(r-1)}{n-k}$ repair bandwidth than traditional MSR codes. A summary is presented in Table 14. It can be seen that our scheme use less storage nodes than the traditional method since each storage node is responsible for storing multiple data blocks. It is often desirable for most practical distributed storage systems where the number of data blocks per node is much greater than one.

COMPARISON OF MINIMUM SPREAD NON-HOMOGENEOUS MODEL VS. TRADITIONAL MODEL BASED ON (n, k) MDS CODES WHERE $n = (h-1)(n-k)+r$.

Scheme	Node	download cost	repair bandwidth
Min. spread Non-homogeneous	h	$\sum_{i=1}^{h-2} w_i(n-k) + rw_{h-1}$	$\gamma_r = (h-1)r$
Traditional Homogeneous	n	$\sum_{i=1}^{k} w_i$	$\gamma_r = \frac{r(n-1)}{n-k}$

Table 14

Numerical Case Study

[00106] To compare the data availability, we examine a scenario of node online probability where the online probability of the first $(h - 1)$ nodes equal p_1 and greater than the rest $(n - h + 1)$ nodes as shown in Equation 50:

$$p_1 = p_2 = \cdots = p_{h-1} \geq p_h = p_{h+1} = \cdots = p_n = p \quad (50)$$

[00107] The data availability of minimum-spread model $Pr_{non\text{-}homo}$ can be computed as Equation 51.

$$Pr_{non-homo} = p_1^{h-1} + (h-1)pp_1^{h-2}(1-p_1) \tag{51}$$

[00108] Since $0 \leq p \leq p_1 \leq 1$, it can be seen that $Pr_{non-homo}$ becomes smaller when h increases. Therefore, we focus mainly on $h = 3, 4$ and compare with traditional model to show the efficiency of minimum-spread model.

[00109] Let begin with $h = 3$, from (50) the first two nodes will have greater online availability than the rest. Therefore, the data availability of traditional model can be computed as below.

$$\begin{aligned} Pr_{homo} = p_1^2 & \left\{ \sum_{r=k-2}^{n-2} \binom{r}{n-2} p^r (1-p)^{n-2-r} \right\} \\ +2p_1(1-p_1) & \left\{ \sum_{r=k-1}^{n-2} \binom{r}{n-2} p^r (1-p)^{n-2-r} \right\} \\ +(1-p_1)^2 & \left\{ \sum_{r=k}^{n-2} \binom{r}{n-2} p^r (1-p)^{n-2-r} \right\} \end{aligned} \tag{52}$$

[00110] In the case of minimum-spread model $n = k + 3$, (52) becomes Equation 53:

$$\begin{aligned} Pr_{homo} = & p^{k+1} + (k+1)p^k(1-p) + p_1 k(k+1)p^{k-1}(1-p)^2 \\ & + \frac{k(k+1)}{2} p_1^2 (1-p)^2 p^{k-2} \left(\frac{k-1}{3} - p\right) \end{aligned} \tag{53}$$

[00111] In order to confirm the increase of data availability, we run some simulations on $h = 3, k = 5$ and $0.5 \leq p \leq p_1 \leq 1$. From Fig. 15, it can be seen that our model can improve the data availability to more than 10% in comparison to the traditional model.

[00112] For checking the efficiency of our scheme on minimum download cost and repair bandwidth, some (n,k) MDS codes are used such that $h = 3, 4$ and $r = 2,3,\cdots,5$. For example, the $(n = 8, k = 5)$ MSR code in the previous example can be used for the case of $(h = 3, r = 2)$.

[00113] From Table 15, it can be seen that our scheme can achieve a smaller download cost and repair bandwidth than the traditional MSR codes. When r increases, the repair

bandwidth can be much smaller than the traditional optimal bandwidth bound, e.g. when r = 5, the repair bandwidth of our scheme is reduced more than 20% of the traditional method. However, our scheme cannot repair any concurrent multi-node failures. Therefore, it is a tradeoff between download cost, repair bandwidth and ability of correcting multi-node failures. Our scheme is best fit with any storage systems with only single-node failure.

COMPARISON OF MINIMUM SPREAD NON-HOMOGENEOUS MODEL VS. TRADITIONAL HOMEGENEOUS MODEL BASED ON (n, k) MSR CODES WHERE $n = (h-1)(n-k)+r$ AND $w_1 \leq w_2 \leq \cdots \leq w_n$. SMALLER VALUE OF DOWNLOAD COST AND REPAIR BANDWIDTH MEANS BETTER PERFORMANCE.

				Allocation	Download cost C_{do}		Repair bandwidth γ_r	
	n	k	r	$(x_1, \cdots, x_k)$	Non-homo.	Traditional	Non-homo.	Traditional
$h=3$	8	5	2	(3, 2, 3)	$3w_1 + 2w_2$	$w_1 + \cdots + w_5$	4	4.67
	11	7	3	(4, 3, 4)	$4w_1 + 3w_2$	$w_1 + \cdots + w_7$	6	7.50
	14	9	4	(5, 4, 5)	$5w_1 + 4w_2$	$w_1 + \cdots + w_9$	8	10.40
	17	11	5	(6, 5, 6)	$6w_1 + 5w_2$	$w_1 + \cdots + w_{11}$	10	13.33
$h=4$	11	8	2	(3, 3, 2, 3)	$3w_1 + 3w_2 + 2w_3$	$w_1 + \cdots + w_8$	6	6.67
	15	12	3	(4, 4, 3, 4)	$4w_1 + 4w_2 + 3w_3$	$w_1 + \cdots + w_{12}$	9	10.50
	19	14	4	(5, 5, 4, 5)	$5w_1 + 5w_2 + 4w_3$	$w_1 + \cdots + w_{14}$	12	14.40
	23	17	5	(6, 6, 5, 6)	$6w_1 + 6w_2 + 5w_3$	$w_1 + \cdots + w_{17}$	15	18.33

Table 15

[00114] Thus it can be seen that three robust and advantageous methodologies related to super-node distributed storage system (DSS) have been provided within two DSS architectures which dynamically select the file fragment size according to the encoding scheme, storage server bandwidth, and file content type. A non-MDS DSS system is provided which has the advantages of: fast and simple encoding over a field of size two, toleration of a maximum of C nodes of failure (where C is a design parameter), repair degree of 2 (i.e. every failed node can be repaired by downloading information from a pair of surviving nodes called a repair pair), C repair pairs for every node, and a low update complexity of C+ 1 (i.e. whenever a piece of information is being updated, only the storage on C+ 1 nodes need to be updated). Further, a non-homogenous DSS is provided which has the advantages of: requiring a smaller file fragment (e.g., four times smaller) and a smaller operational field size when use with (k+2, k) MDS code, while maintaining the same optimal repair bandwidth and total storage as other homogenous (k+2, k) DSSs; and achieving a lower repair bandwidth for a 1-failure scenario when relaxing the MDS

property. Besides super-node scheme, our minimum-spread scheme can achieve the minimum download cost and require lower repair bandwidth of r-failures by r(r- 1)M/(n- k)k than the optimal bandwidth bound in the traditional homogeneous DSS.

Local Repairable Code DSS

[00115] In a distributed storage system, a data file is stored at a distributed collection of storage devices/nodes in a network. Since any storage device is individually unreliable and subject to failure (i.e. erasure), redundancy must be introduced to provide the much-needed system-level protection against data loss due to device/node failure. The simplest form of redundancy is *replication*. By storing c identical copies of a file at c distributed nodes, one copy per node, a c-replication system can guarantee the data availability as long as no more than $(c-1)$ nodes fail. Such systems are very easy to implement, but extremely inefficient in storage space utilization, incurring tremendous waste in devices and equipment, building space, and cost for powering and cooling. More sophisticated systems employing erasure coding can expect to considerably improve the storage efficiency. Consider a file that is divided into k equal-size fragments. A judiciously-designed $[n,k]$ erasure (systematic) code can be employed to encode the k data fragments (termed systematic symbols in the coding jargon) into n fragments (termed coded symbols) stored in n different nodes. If the $[n,k,d]$ code reaches the Singleton bound such that the minimum Hamming distance satisfies $d = n - k + 1$, then the code is maximum distance separable (MDS) and offers redundancy-reliability optimality. With an $[n,k]$ MDS erasure code, the original file can be recovered from any set of k encoded fragments, regardless of whether they are systematic or parity. In other words, the system can tolerate up to $(n - k)$ concurrent device/node failure without jeopardizing the data availability.

[00116] Despite the huge potentials of MDS erasure codes, however, practical application of these codes in massive storage networks have been difficult. Not only are simple (i.e.

requires very little computational complexity) MDS codes very difficult to construct, but data repair would in general require the access of k other encoded fragments, causing considerable input/output (I/O) bandwidth that would pose huge challenges to a typical storage network.

[00117] Motivated by the desire to reduce repair cost in the design of erasure codes for distributed storage systems, the notion of symbol locality was introduced in linear codes. The ith coded symbol of an $[n,k]$ linear code C is said to have locality r ($1 \leq r \leq k$) if it can be recovered by accessing at most r other symbols in C. The concept was further generalized to (r,δ) locality, to address the situation of multiple device failures.

[00118] The ith code symbol $c_i, 1 \leq i \leq n$, in an $[n,k]$ linear code C is said to have locality (r,δ) if there exists an index set $S_i \subseteq [n]$ containing i such that $|S_i|-\delta+1 \leq r$ and each symbol $c_j, j \in S_i$, can be reconstructed by any $|S_i|-\delta+1$ symbols in $\{c_l; l \in S_i \text{ and } l \neq j\}$, where $\delta \geq 2$ is an integer. Thus, when $\delta = 2$, the notion of (r,d) locality reduces to the notion of r locality. Two cases of (r,δ) codes are introduced in the literature: An $(r,\delta)_i$ code is a systematic linear code whose information symbols all have locality (r,δ); and an $(r,\delta)_a$ code is a linear code all of whose symbols have locality (r,δ). Hence, an $(r,\delta)_a$ code is also referred to as having all-symbol locality (r,δ), and an $(r,\delta)_i$ code is also referred to as having *information* locality (r,δ). A symbol with (r,δ) locality – given that at most $(\delta-1)$ symbols are erased – can be deduced by reading at most r other unerased symbols.

[00119] Clearly, codes with a low symbol locality, such as $r < k$, impose a low I/O bandwidth and repair cost in a distributed storage system. In a DSS system, one can use "group" to describe storage nodes situated in the same physical location which enjoy a higher communication bandwidth and a shorter communication distance than storage nodes belonging to different groups. In the case of node failure, a locally repairable code makes it possible to efficiently recover data stored in the failed node by downloading

information from nodes in the same group (or in a minimal number of other groups). Fig. 16 provides a simple example of how an $(r,\delta)_a$ code is used to construct a distributed storage system. In this example, C is a$(2,3)_a$ linear code of length 12 and dimension 5. Note that a failed node can be reconstructed by accessing only two other existing nodes, while it takes five existing nodes to repair a failed node if a [12,5] MDS code is used.

[00120] Locality was identified as a repair cost metric for distributed storage systems by introducing the concept of symbol locality of linear codes and establishing a tight bound for the redundancy in terms of the message length, the distance, and the locality of information coordinates. For a generalized concept, i.e., (r,δ) locality, the minimum distance d of an $(r,\delta)_i$ linear code C is upper bounded by Equation 54:

$$d \leq n - k + 1 - \left(\left\lceil \frac{k}{r} \right\rceil - 1 \right) (\delta - 1) \tag{54}$$

where n and k are the length and dimension of C respectively. A class of codes known as pyramid codes may achieve this bound. Since an $(r,\delta)_a$ code is also an $(r,\delta)_i$ code, (54) also presents an upper bound for the minimum distance of $(r,\delta)_a$ codes.

[00121] An $(r,\delta)_a$ code (systematic or not) is also termed a locally repairable code (LRC), and $(r,\delta)_a$ codes that achieve the minimum distance bound are called optimal. There exists optimal locally repairable linear codes when $(r + \delta - 1)|n$ and $q > kn^k$.

[00122] Under the condition that $(r+\delta - 1)|n$, a construction method of optimal locally repairable vector codes was proposed, where maximal rank distance (MRD) codes were used along with MDS array codes. For the special case of $\delta = 2$, it was proposed an explicit construction of optimal LRCs when

$$(r+1)|n$$

or

$$n \bmod (r+1) - 1 \geq k \bmod r > 0$$

[00123] Except for the special case that $n \bmod (r+1)-1 \geq k \bmod r > 0$, no results are known about whether there exists optimal $(r,\delta)_a$ code when $(r+\delta-1) \nmid n$. Up to now, designing LRCs with optimal distance remains an intriguing open problem for most coding parameters n,k,r and δ. Since large fields involve rather complicated and expensive computation, a related interesting open problem asks how to limit the design (of optimal LRCs) over relatively smaller fields.

Algorithm to construct optimal $(r,\delta)_a$ linear codes

[00124] We investigate the structure properties and the construction of optimal $(r,\delta)_a$ linear codes of length n and dimension k. A simple property of optimal $(r,\delta)_a$ linear codes is proved, which shows that $\frac{n}{r+k-1} \geq \frac{k}{r}$ for any optimal $(r,\delta)_a$ linear code. Hence we impose this condition of $\frac{n}{r+k-1} \geq \frac{k}{r}$ throughout our discussion of optimal $(r,\delta)_a$ codes.

[00125] We prove a structure theorem for the optimal $(r,\delta)_a$ linear codes for $r|k$. This structure theorem indicates that it is possible for optimal $(r,\delta)_a$ linear codes, a sub-class of optimal $(r,\delta)_i$ linear code, to have a simpler structure than otherwise.

[00126] We prove that there exist no optimal $(r,\delta)_a$ linear codes for

$$(r+\delta-1) \nmid n \text{ and } r|k \tag{55}$$

or

$$m < v+\delta-1 \text{ and } u \geq 2(r-v)+1 \tag{56}$$

where $n = w(r+\delta-1)+m$ and $k = ur+v$ such that $0 < v < r$ and $0 < m < r+\delta-1$.

[00127] We propose algorithm 1 for constructing optimal $(r,\delta)_a$ linear codes over any field of size $q \geq \binom{n}{k-1}$ when

$$(r+\delta-1) | n \tag{57}$$

or

$$m \geq v+\delta-1 \tag{58}$$

where $n = w(r+\delta-1)+m$ and $k = ur+v$ such that $0 < v < r$ and $0 < m < r+\delta-1$.

[00128] We propose algorithm 2 for constructing optimal $(r,\delta)_a$ linear codes over any field of size $q \geq \binom{n}{k-1}$ when

$$w \geq r+\delta-1-m \text{ and } r-v \geq u \tag{59}$$

or

$$w+1 \geq 2(r+\delta-1-m) \text{ and } 2(r-v) \geq u \tag{60}$$

where $n = w(r+\delta-1)+m$ and $k = ur+v$ such that $0 < v < r$ and $0 < m < r+\delta-1$.

[00129] A summary of our results is given in Fig 17. Note that if none of the conditions in (55)-(58) holds, it then follows that

$$m < v+\delta-1 \text{ and } u \leq 2(r-v).$$

[00130] In that case, if condition (59) does not hold, we have $w < r+\delta-1-m$ or $r-v < u$; and if condition (60) does not hold, we have $w+1 < 2(r+\delta-1-m)$, i.e., $w < 2(r+\delta-1-m)-1$. Hence, if, neither condition (59) nor condition (60) holds (in addition to (55)-(58)), then one of the following two conditions must be satisfied:

$$w < r+\delta-1-m, \tag{61}$$

or

$$r+\delta-1-m \leq w < 2(r+\delta-1-m)-1 \text{ and } r-v < u \tag{62}$$

[00131] In other words, if none of the conditions (55)-(60) holds, then either (61) or (62) will hold. From our existence proof and/or constructive results, the existence of optimal $(r,\delta)_a$ linear code is not known only for a limited scope with parameters described by (61) and (62).

[00132] We develop the following generic method to construct optimal $(r,\delta)_a$ linear codes. In this method, a generating matrix $G = (G_1,\cdots,G_n)$ of the optimal code C is constructed in four steps. In the first step, construct a collection of index sets, say, $S_1,\ldots,S_t$, which are the candidates of indices of local MDS code. In the second step, select a subset from each index sets of step 1 to form one index set which is the candidate of indices of the largest MDS code contained in the final optimal $(r,\delta)_a$ linear codes. In the third step, build

a sub-matrix G^0, consisting of a subset of columns of G, such that G^0 is a generating matrix of a maximum distance separable (MDS) code. In the final step, construct the other columns of G, one by one, such that some special property is satisfied.

[00133] In the following we describe two specific algorithms that follow the above method to construct the optimal $(r,\delta)_a$ linear codes shown in Figure 17.

Algorithm 1

[00134] If $q \geq \binom{n}{k-1}$, then we can construct an optimal $(r,\delta)_a$ linear code of length n and dimension k over F_q such that for each subset S_i and each code symbol c_i in S_i, c_i can be reconstructed by any $|S_i|$-δ+1 other symbols in S_i. Specifically, we have the following construction. A collection of subsets of indexes S = $\{S_1,\cdots,S_t\}$. E,g, S = {{1,2,3}, {4,5,6}, {7,8,9}, {10,11,12}, {13,14}} are constructed where two conditions must be satisfied. These subsets are disjoint and their union is the full set of indexes. The union of any $\lceil \frac{k}{r} \rceil$ subsets contains at least $k + \lceil \frac{k}{r} \rceil(\delta - 1)$ indexes. A subset $S \subseteq [n]$ is called an (S,r)-*core* if S intersect each S_i with at most $|S_i| - \delta + 1$. Additionally, if S is an (S,r)-core and S contains k indexes, then S is called an (S,r,k)-*core*.

[00135] A subset is picked from each subset of indexe S_i and formed into a subset by taking union. E.g. pick {1,2},{5,6},{7,8},{10,12},{13} and form {1,2,5,6,7,8,10,12,13}. In general, from each subset S_i, pick a subset $U_i \subseteq S_i$ such that U_i contains $|S_i|-\delta+1$ indexes. Then form a set Ω_0 by taking union, i.e., the set Ω_0 is the union of all U_is.

[00136] A generator matrix for MDS code of size $|\Omega_0|$ is constructed which is used as the column of the final generator matrix, where the columns are indexed by the set Ω_0, e.g. the third column of the MDS generator matrix will be in the fifth column of the final generator matrix. E.g. $\{G_1,G_2,G_5,G_6,G_7,G_8,G_{10},G_{12},G_{13}\}$, where each G_i is a column of the generator matrix.

[00137] Fill in other columns of G according to certain properties. This can be achieved by the following process:

1 Let $\Omega = \Omega_0$;
2 i runs from 1 to t;
3 While $S_i \backslash \Omega \neq \emptyset$
4 Pick a $\lambda \in S_i \backslash \Omega$ and let Λ be the set of all $S_0 \subseteq \Omega$ such that $S_0 \cup \{\lambda\}$ is an (S, r, k)- core;
5 Let $G_\lambda \in <\{G_i\}_{i \in S_i \cap \Omega}> \backslash (\cup_{S_0 \in \Lambda} <\{G_i\}_{i \in S_0}>)$;
6 $\Omega = \Omega \cup \{\lambda\}$;

[00138] By the above process, we can obtain a matrix $G = (G_1, \cdots, G_n)$ over F_q. Let C be the linear code generated by G. Then C is an optimal $(r,\delta)_a$ linear code over F_q. When $(r+\delta-1)|n$ or $n \bmod (r+\delta-1)-(\delta-1) \geq k \bmod r > 0$, we can construct an optimal $(r,\delta)_a$ linear code C of length n and dimension k over F_q by the above method, where $q \geq \binom{n}{k-1}$. For the case of $(r+\delta-1)|n$, we simply let $\{S_1, \cdots, S_t\}$ be a partition of $[n]$ such that $|S_i| = r+\delta-1, \forall i \in [t]$, where $t = \frac{n}{r+\delta-1}$. For the case of $n \bmod (r+\delta-1)-(\delta-1) \geq k \bmod r > 0$, we let $\{S_1, \cdots, S_t\}$ be a partition of $[n]$ such that $|S_i| = r+\delta-1$ for $i \in \{1, \cdots, t-1\}$ and $|S_t| = n \bmod (r+\delta-1)$. Then use the above method, we can construct an optimal $(r,\delta)_a$ linear code over F_q.

Algorithm 2

[00139] If $q \geq \binom{n}{k-1}$, then we can construct an optimal $(r,\delta)_a$ linear code of length n and dimension k over F_q such that for each subset S_i and each code symbol c_i in S_i, c_i can be reconstructed by any r other symbols in S_i. Specifically, we have the following construction. A collection of subsets of indexes S = $\{S_1, \cdots, S_t\}$ is constructed. The indexes can be overlapping. E.g., S = {{1,2,3,4,5}, {1,6,7,8,9}, {1,10,11,12,13} }, {{14,15,16,17,18}, {14,19,20,21,22}}, {{23,24,25,26,27}, {28,29,30,31,32}, {33,34,35,36,37}}.

[{1,2,3,4,5},{1,6,7,8,9},{1,10,11,12,13}] ←– one cluster, common point is 1.

[{14,15,16,17,18},{14,19,20,21,22}] ←– one cluster, common point is 14.

[{23,24,25,26,27}] ←– one cluster. [{28,29,30,31,32}] ←– one cluster. [{33,34,35,36,37}] ←– one cluster. This example is shown in Fig 18.

[00140] Numbers in any two clusters must be disjoint, within the same cluster, all subsets need to have exactly only one common point, and different cluster can have different number of subset. In general, there is a collection A = $\{A_1,\cdots,A_\alpha,B\}$ which is a partition of $[t]$ and a subset $\Psi = \{\xi_1,\cdots,\xi_\alpha\} \subseteq [n]$ such that the following conditions are satisfied: for each $j \in [\alpha]$, $\{\xi_j\} = \cap_{i\in A_j} S_i$ and $\{S_i \backslash \{\xi_j\}; i \in A_j\}$ are mutually disjoint; $\{\cup_{i\in A_j} S_i;\ j \in [\alpha]\} \cup \{S_j;\ j \in B\}$ is a partition of $[n]$; the union of any $\lceil \frac{k}{r} \rceil$ subsets contains at least $k + \lceil \frac{k}{r} \rceil(\delta - 1)$ indexes.

[00141] A subset $S \subseteq [n]$ is said to be an (S,r)-*core* if the following three conditions hold: If $j \in [\alpha]$ and $\xi_j \in S$, then $|S \cap S_i| \leq r, \forall i \in A_j$; if $j \in [\alpha]$ and $\xi_j \notin S$, then there is an $i_j \in A_j$ such that $|S \cap S_{i_j}| \leq r$ and $|S \cap S_i| \leq r-1, \forall i \in A_j \backslash \{i_j\}$; if $i \in B$, then $|S \cap S_i| \leq r$. Additionally, if $S \subseteq [n]$ is an (S,r)-core and $|S| = k$, then S is called an (S,r,k)-*core*.

[00142] Then a subset is selected from each subset of indexes S_i and form a subset by taking union. E.g. pick {1,2,3},{1,6,7},{1,10,11},{14,15,16},{14,19,20},{23,24,25},{28,29,30}, {33,34,35} and form {1,2,*3*,6,7,10,*11*,14,15,16,,19,20,23,24,25,28,29,30,33,34,35}. In general, for each $j \in [\alpha]$ and $i \in A_j$, pick a $U_i \subseteq S_i$ such that $|U_i| = r$ and $\xi_j \in U_i$. For each $i \in B$, pick a $U_i \subseteq S_i$ such that $|U_i| = r$. Let $\Omega_0 = \cup_{i\in[t]} U_i$.

[00143] A generator matrix for MDS code of size $|\Omega_0|$ is constructed which is used as the columns of the final generator matrix, where the columns are indexed by the set Ω_0, e.g. the fourth column of the MDS generator matrix will be in the sixth column of the final generator matrix. E.g. (G_1,G_2,G_3,G_6,G_7,G_{10},G_{11}, G_{14},G_{15},G_{16},G_{19}, G_{20}, G_{23}, G_{24}, G_{25}, G_{28}, G_{29}, G_{30}, G_{33}, G_4, G_{35}), where each G_i is a column of the generator matrix.

[00144] Fill in other columns according to certain properties. This can be achieved by the following process:

1 Let $\Omega = \Omega_0$;
2 i runs from 1 to t;
3 While $S_i \backslash \Omega \neq \emptyset$
4 Pick a $\lambda \in S_i \backslash \Omega$ and let Λ be the set of all $S_0 \subseteq \Omega$ such that $S_0 \cup \{\lambda\}$ is an (S,r,k)- core;
5 Let $G_\lambda \in <\{G_l\}_{l \in S_i \cap \Omega}> \backslash (\cup_{S_0 \in \Lambda} <\{G_l\}_{l \in S_0}>)$;
6 $\Omega = \Omega \cup \{\lambda\}$;

[00145] By Algorithm 2, we can obtain a matrix $G = (G_1, \cdots, G_n)$ over F_q. Let C be the linear code generated by G. Then C is an optimal $(r,\delta)_a$ linear code over F_q. When $0 < n \bmod (r + \delta - 1) < k \bmod r + (\delta - 1)$, with other additional conditions, we can construct an optimal $(r,\delta)_a$ linear code C of length n and dimension k over F_q by the above method, where $q \geq \binom{n}{k-1}$.

[00146] For example, suppose $w \geq r + \delta - 1 - m$ and $r - v \geq u$, where $n = w(r + \delta - 1) + m$ and $k = ur + v$ such that $0 < m < r + \delta - 1$ and $0 < v < r$. We let $\{S_1, \cdots, S_t\}$ be a collection of $(r + \delta - 1)$- subsets of $[n]$ such that $S_1, \cdots, S_{l+1}$ have a common element and $\cup_{i=1}^{t} S_i = [n]$, where $l = r + \delta - 1 - m$. Then use the above method, we can construct an optimal $(r,\delta)_a$ linear code over F_q.

[00147] Suppose $w+1 \geq 2(r+\delta-1-m)$ and $2(r-v) \geq u$, where $n = w(r+\delta-1)+m$ and $k = ur+v$ such that $0 < m < r + \delta - 1$ and $0 < v < r$. We let $\{S_1, \cdots, S_t\}$ be a collection of $(r + \delta - 1)$-subsets of $[n]$ such that S_{2i-1} and S_{2i} have a common element $(i = 1, \cdots, l)$ and $\cup_{i=1}^{t} S_i = [n]$, where $l = r+\delta-1-m$. Then use the above method, we can construct an optimal $(r,\delta)_a$ linear code over F_q.

[00148] Thus it can be seen that systems and methods for distributed storage systems that are highly recoverable and relatively impervious to failure have been disclosed which provide many advantages. While several exemplary embodiments have been presented in the foregoing detailed description of the invention, it should be appreciated that a number

of variations may exist, including variations as to the system structure, operation and methodologies of the distributed storage systems.

[00149] It should further be appreciated that the exemplary embodiments are only examples, and are not intended to limit the scope, applicability, dimensions, or configuration of the invention. Rather, the foregoing detailed description will provide those skilled in the art with a convenient road map for implementing an exemplary embodiment of the invention, it being understood that various changes may be made in the systems and methodologies of the exemplary embodiments without departing from the scope of the invention as claimed.

Claims

1. A distributed storage system DSS comprising

h non-empty nodes; and

data stored non-homogenously across the non-empty nodes according to the storing codes (n,k).

2. The system in claim 1 wherein the *h* non-empty nodes each having respective non-homogenous bandwidths.

3. The system in claim 1 or 2 wherein one of the non-empty nodes is a super-node with a significantly higher bandwidth, reliability and/or storage capacity than the remaining non-empty nodes, and a significantly higher proportion of the data is stored on the super-node.

4. The system in claim 3 wherein the super-node is a local host.

5. The system in claim 3 wherein the super-node is configured to store two or more systematic data sub-blocks using Maximum Distance Separable Coding.

6. The system in claim 3 wherein the super-node is configured to store two or more systematic data sub-blocks using Non Maximum Distance Separable Coding.

7. The system in claim 3 wherein the super-node is configured to store two or more parity data sub-blocks using Maximum Distance Separable Coding.

8. The system in claim 1 or 2 configured to optimise the distribution of data across the non-empty nodes to minimise the repair bandwidth and/or the download cost.

9. The system in claim 8 where h non-empty nodes store the same amount of information.

10. The system in claim 8 where h-1 non-empty nodes store the same amount of information.

11. The system in claim 1 or 2 further comprising a plurality of empty nodes and the method further comprising minimising h.

12. A method for determining linear erasure codes with local repairability comprising

selecting two or more coding parameters including r and δ;

determining if an optimal [n, k, δ] code having all-symbol (r, δ)-locality ("(r, δ)$_a$") exists for the selected r, δ; and

if the optimal (r, δ)$_a$ code exists performing a local repairable code using the optimal (r, δ)$_a$ code.

13. The method in claim 12 wherein the coding parameters further including n and k.

14. The method in claim 13 further comprising determining the lower bound of the required field size.

15. The method in any of claims 12 to 14 wherein when the coding parameters satisfy

$a)\ w \geq r + \delta - 1 - m$ and $r - v \geq u$

b) $w + 1 \geq 2(r + \delta - 1 - m)$ and $2(r - v) \geq u$

c) $r + \delta - 1) \mid n$, or

d) $m \geq (v + \delta - 1)$

an optimal (r, δ)$_a$ code exists.

16. The method in claim 15 further comprising determining an optimal (r, δ)$_a$ code using a first algorithm for (a) and (b) and a second algorithm for (c) and (d).

17. The method in claim 14 wherein the lower bound is determined using $\binom{n}{k-1}$.

18. The method in any of claims 12 to 18 wherein when the coding parameters satisfy

e) $(r + \delta - 1) \nmid n$ and $r|k$, or

f) $m < v + \delta - 1$ and $u \geq 2(r - v) + 1$

no optimal (r, δ)$_a$ code exists.