https://doi.org/10.7840/kics.2016.41.12.1684

다양한 도움 노드의 수를 가지는 재생 부호의 설계

이 혁•, 이 정 우°

The Design of Regenerating Codes with a Varying Number of Helper Nodes

Hyuk Lee•, Jungwoo Lee°

요 약

최근 분산 저장 시스템에 erasure code를 활용하여 저장소 효율성을 높이려는 연구가 활발히 진행되고 있다. 재생 부호(regenerating codes)는 erasure code의 일종으로, 높은 저장소 효율성과 네트워크 효율성을 가지는 코드이다. (n,k,d)-재생 부호는 n개의 저장소 노드를 가지며, 손실된 노드가 발생하였을 때, 해당 노드는 d개의 살아남은 노드로부터 정보를 다운로드받아 복구될 수 있다. 하지만 일반적인 재생 부호는 노드 복구 시 정확히 d개의 도움 노드들을 사용해야 하며, 노드 손실이 빈번하거나, 노드 간 접속이 불안정한 환경에서, d개 이하의 노드들에만 접속 가능할 경우에 유연하게 대처할 수 없다. 본 논문에서는 약간의 복구 대역폭의 희생을 통하여, $k \leqq \bar{d} \leqq d$의 다양한 도움 노드의 수 $\bar{d}$개로 노드를 복구할 수 있는 유연한 코드 운용 방식을 제안하였다.

Key Words : Regenerating codes, Distributed storages, Repair bandwidth, Network coding, Erasure codes

ABSTRACT

Erasure codes have recently been applied to distributed storage systems due to their high storage efficiency. Regenerating codes are a kind of erasure codes, which are optimal in terms of minimum repair bandwidth. An (n,k,d)-regenerating code consists of n storage nodes where a failed node can be recovered with the help of the exactly d numbers of surviving nodes. However, if node failures occur frequently or network connection is unstable, the number of helper nodes that a failed node can contact may be smaller than d. In such cases, regenerating codes cannot repair the failed nodes efficiently since the node repair process of the codes does not work when the number of helper nodes is less than d. In this paper, we propose an operating method of regenerating codes where a failed node can be repaired from $\bar{d}$ helper nodes where $k \leqq \bar{d} \leqq d$.

Ⅰ. 서 론

최근 SNS나 클라우드 서비스의 사용량 증가와 더불어, 대규모의 데이터를 네트워크상에 효율적이고 안정적으로 저장할 수 있는 분산 저장 시스템(distributed storage system)에 대한 연구가 활발하게 진행되고 있다. 하둡 분산 파일 시스템(Hadoop distributed file system)[1]이나, Windows Azure

※ 본 연구는 한국연구재단 중견연구자지원사업(NRF-2015R1A2A1A15052493), 산업통상자원부 산업기술혁신사업(10051928), 방위사업청 국방생체모방자율로봇특화센터(UD130070ID), BK21플러스 창의정보기술 인재양성사업단 및 뉴미디어통신공동연구소 지원으로 수행되었습니다.

• First Author : Department of Electrical and Computer Engineering, INMAC, Seoul National University, hyuklee@wspl.snu.ac.kr, 학생회원

° Corresponding Author : Department of Electrical and Computer Engineering, INMAC, Seoul National University, junglee@snu.ac.kr, 종신회원

논문번호 : KICS2016-10-296, Received October 7, 2016; Revised December 6, 2016; Accepted December 6, 2016

Storage[2]를 비롯한 분산 저장 시스템은 대규모의 데이터 파일을 네트워크로 연결된 다수의 노드에 분산적으로 저장하는 시스템을 말한다.

여러 개의 저장소 노드에 데이터를 분산적으로 저장할 경우, 시스템에 접근하고자 하는 각 사용자의 상황에 따라 각 노드에 대한 접근성이 달라질 수 있다. 데이터에 접근하고자 하는 유저(data collector)들이 전체가 아닌 일부의 노드에만 접속하여도 전체의 파일을 복원할 수 있는 것이 분산 저장 시스템의 기본적인 기능이다. 총 n개의 노드들 중 임의의 k개에만 접속하여도 전체 파일을 복원할 수 있는 성질을 (n,k)-복원 속성((n,k)-recoverability property)이라고 부른다.

한편, 많은 데이터 디스크 드라이브로 이루어진 분산 저장 시스템을 운용할 경우, 일부의 노드들이 손실되는 경우가 빈번하게 발생한다. 일부 노드들의 고장으로 인한 데이터 손실에 대비하기 위하여, 살아남은 노드들로부터 정보를 전송받아 고장난 노드를 복구하는 노드 복원 속성(node repair propoety) 또한 분산 저장 시스템의 필수적인 기능이다.

이러한 기능들을 충족시키기 위하여, 특정 량의 반복되는(redundant) 데이터를 시스템에 저장시켜 놓을 필요가 있다. 일반적으로 과거의 분산 저장 시스템에서는 이러한 반복 데이터를 얻기 위해 노드 복제(replication) 방식이 주로 사용되어 왔다. 예를 들어 하둡 분산 파일 시스템(Hadoop distributed file system)에서는 노드 손실 및 일시적 접속 불량에 대비하기 위해 같은 정보를 가지는 노드들을 3개씩 저장하는 3-way replication 방식을 활용한다. 하지만 단순히 복제된 데이터를 반복 저장하는 방식은 저장 공간의 낭비를 발생시킨다. 이러한 문제를 해결하기 위하여, 최근 분산 저장 시스템에 erasure coding 기법을 적용하여 반복 데이터를 효율적으로 활용하기 위한 방법들이 활발히 연구되고 있다. 실제 예로서, 페이스북에서는 erasure coding의 일종인 [14,10] Reed Solomon code를 활용하여 데이터를 저장하는 방식을 활용한다. 이 경우, 기존의 replication 방식에 비하여 65%의 저장 용량을 절약할 수 있음이 알려져 있다[3].

어떤 노드가 손실될 경우, 이는 신규 노드(newcomer)라고 부르는 새로운 노드로 대체되어야 하며, 신규 노드가 생성되기 위해서는 도움 노드(helper node)라고 불리는 살아남은 노드들로부터 정보를 다운로드받아야 한다. 이 때, 손실된 노드의 복구를 효율적으로 진행하는 것 또한 분산 저장 시스템에 관한 연구에서의 중요한 과제이다. 노드 복구의 효율성 증대 방안은 크게 대표적인 두 가지의 방향으로 연구되고 있다. 복구 시 접속해야 하는 도움 노드의 수를 뜻하는 부분 접속수(locality)를 최적화하는 locally recoverable codes[4-7], 그리고 한 개의 신규 노드를 생성하기 위해 필요한 데이터 다운로드 양인 복구 대역폭(repair bandwidth)을 최적화하는 재생 부호(regenerating codes)[8,9]가 그것이다.

전통적인 erasure coding 방식은 저장 용량 효율성 측면에서 강점을 갖는 반면, 높은 복구 대역폭을 가진다는 점에서 네트워크 효율성 측면에서 약점을 가진다. 예를 들어, α만큼의 용량을 가지는 노드가 손실된 상황을 생각해 보자. 노드 복제 방식에서는 손실된 노드와 같은 정보를 가지는 한 개의 노드로부터 데이터를 복사해 오면 되므로, α만큼의 복구 대역폭이 필요하다. 하지만 RS 코드와 같은 (n,k)-erasure code의 경우 $k\alpha$만큼의 복구 대역폭을 필요로 한다.

Dimakis 등은 적은 양의 복구 대역폭을 가지는 새로운 종류의 erasure code를 제안하였으며, 각 노드별 저장용량 α와 복구 대역폭 $\gamma(=d\beta)$ 간의 최적의 상충관계를 제시하였다[8]. 이러한 저장량-통신량 간의 상충관계를 만족하는 코드들을 재생 부호라고 부른다. (n,k,d)-regeneraitng code는 B크기의 데이터를 각각 α만큼의 용량을 갖는 n개의 저장소 노드들에 분산적으로 저장하며, 다음의 두 성질을 만족한다.

(1) (n,k)-복원 속성 - n개의 노드 중 임의의 k개 노드에 접속하면 B 크기의 전체 데이터를 복원할 수 있다.

(2) 노드 복원 기능 - 한 개의 손실된 노드는 나머지 $n-1$개의 노드들 중, 임의의 d개의 살아남은 노드(helper node)들로부터 $\beta(\leq \alpha)$ 만큼씩의 데이터를 다운로드 받아 복구될 수 있다.

또한, [8]에서는 위의 두 속성을 가지는 재생 부호를 이용한 분산 저장 시스템에 저장 가능한 총 데이터량 B의 상한을 주어진 노드 별 저장량 α와 도움 노드가 한 신규 노드에게 보내는 데이터량 β값으로 이루어진 식으로 도출하였는데, 이는 다음의 식 (1)과 같다.

$$B \leq \sum_{i=1}^{k} \min(\alpha, (d-i+1)\beta). \qquad (1)$$

위의 식에 표현된 B의 상한은 재생 부호의 erasure code로서의 기능인 (n,k)-복원 속성과 고유의 노드

복원 기능이 연속적으로 사용되는 정보 흐름 그래프(information flow graph)로 표현하여, 이의 네트워크 코딩 이론에 기반하여 도출되었으며, 또한 이 상한을 등호로 만족하는 코딩 기법이 실제로 존재함도 증명하였다.

위의 식 (1)을 등호로 성립하게 하는 (α,β) 쌍들의 집합을 구해 보면, 어떤 (α,β)도 다른 (α,β)에 비해 α, β가 동시에 작을 수 없음을 알 수 있다. 이러한 점들을 $\alpha-\beta$ 평면 상에 나타냈을 때, 부분적으로 선형인 형태(piece-wise linear)를 가지며, 저장량-통신량 상충 관계(storage bandwidth tradeoff)를 형성한다. 이 점들 위에 존재하는 (α,β)점 중, 가장 작은 α값을 가지는 경우를 MSR (minimum storage regenerating) code, 가장 작은 β값을 가지는 경우를 MBR (minimum bandwidth regeneraitng) code 라고 부르며, 다음과 같은 α, β값을 갖는다.

$$(\alpha_{MSR},\beta_{MSR})=\left(\frac{B}{k},\frac{B}{k(d-k+1)}\right)$$
$$(\alpha_{MBR},\beta_{MBR})=\left(\frac{2Bd}{k(2d-k+1)},\frac{2B}{k(2d-k+1)}\right)$$

여기서 주목해야 할 점은, $\frac{\alpha}{\beta}$의 값은 항상 $d-k+1\leqq\frac{\alpha}{\beta}\leqq d$의 범위를 가지며, $d\geqq k$일 경우, 항상 $k\alpha\geqq d\beta$를 만족한다는 것이다.

(n,k,d)-재생 부호는 어떠한 한 개의 손실된 노드가 복구될 때, d개의 노드들로부터 도움을 받는 상황에 최적화되어 있다. 하지만, 일시적으로 많은 손실이나 접속 불량이 발생하여 어떠한 고장난 노드가 접속할 수 있는 노드의 수가 d개보다 작을 때는 노드 복원 기능을 사용할 수 없게 되며, 이 때 노드를 복구하기 위한 방법은 (n,k)-복원 속성을 이용해 k개의 노드로부터 α만큼씩의 데이터를 다운로드받아 B크기의 전체 데이터를 복구한 후, 이를 이용해 자신이 필요한 α개의 정보를 다시 부호화하여야 한다. 하지만 이는 복구 대역폭이 $d\beta$에서 $k\alpha$ $(\geqq d\beta)$로 증가함을 의미한다.

본 논문에서는, 노드 복원 시, 도움 노드의 수 d를 자유롭게 조절 가능한 재생 부호의 운용 방법에 대해서 논의한다. 구체적으로, 제안되는 코드 운용방식을 사용하면, 약간의 복구 대역폭의 희생으로, 노드 복원 시 사용 가능한 도움 노드의 수 $\bar{d}$를 k에서 d사이로 조절 가능하다. 더하여, 네트워크 상황에 따라 각각의 도움 노드들이 전송해야 하는 데이터량을 조절하는 것도 가능하다. 제안하는 설계 방식은 임의의 (n,k,d)-재생 부호로부터 도출 가능하며, 임의의 도움 노드 수 $\bar{d}$ $(k\leqq\bar{d}\leqq d)$를 가질 수 있으며, 도움 노드의 수 $\bar{d}$와 복구 대역폭 $\bar{\gamma}$의 관계는 다음의 정리 1과 같이 기술할 수 있다.

정리 1. 임의의 (n,k,d)-재생 부호의 손실된 노드는 $\bar{d}$ $(k\leqq\bar{d}\leqq d)$개의 도움 노드들에 의해 복구될 수 있다. 이 때, 복구 대역폭 $\bar{\gamma}$는 다음과 같다.

$$\bar{\gamma}=d\beta+(d-\bar{d})(k-1)\beta \tag{2}$$

증명. 정리 1의 증명은 II-3절에서 다룬다.■

예를 들어, B의 정보를 저장하는 $(n,k,d)=(10,5,8)$ MSR 코드의 경우, 8개의 도움 노드를 활용할 때, $\gamma_{MSR}=d\beta_{MSR}=\frac{Bd}{k(d-k+1)}=0.4B$의 복구 대역폭을 갖는데, 만약 8개 미만의 도움 노드에만 접속할 수 있는 경우에는 $k\alpha_{MSR}=B$만큼의 복구 대역폭을 필요로 한다. 하지만 위의 정리 1에서 제안한 방식을 사용할 경우, 접속 가능한 도움 노드의 수가 7, 6, 5인 경우 각각 $0.6B$, $0.8B$, B의 복구 대역폭으로 손실된 노드를 복구 가능하므로, 복구 대역폭의 절약이 가능해진다.

Ⅱ. 본 론

(n,k,d,α,β)의 파라미터들을 가지는 어떤 재생 부호가 있다고 가정하자. $1\times B$ 크기의 부호화되지 않은 행벡터 $\overline{m}$은 $1\times n\alpha$ 크기의 행백터 $\bar{c}$로 부호화된다. $\bar{c}$를 순서대로 α개씩 n개의 부분으로 나누었을 때, i번째에 해당하는 α개의 심볼들이 i번째 노드에 저장되는 심볼들이다. 이런 구조로 보았을 때, 부호화 및 복호화 과정이 모두 선형(linear)이라고 가정한다면, 이 재생 부호는 $(n\alpha,B)$-선형 블록 코드의 한 종류라고 할 수 있으며, 다음의 식을 만족하는 $B\times n\alpha$크기의 생성 행렬 G와 $(n\alpha-B)\times n\alpha$ 크기의 패리티 체크 행렬 H를 정의할 수 있다.

$\bar{c} = \overline{m} G,\ GH^T = 0$

논문의 남은 부분에서 우리는 다음의 수식 표현들을 사용할 것이다. $m,\ n$ $(m \leq n)$이 자연수일 때, $[m] = \{1,2,...,m\},\ [m,n] = \{m, m+1, ..., n\}$을 의미하며, 어떤 행렬 M에 대해 $S(M)$은 M의 열공간(column space)을, $S(M^T)$는 행공간(row space)를 각각 의미한다. 그리고 $n \times n$크기의 항등행렬을 I_n으로 나타낼 것이다. 또한 $|M|$은 집합 M의 원소의 개수를 의미한다.

2.1 선형블록코드의 성질을 이용한 보조 정리들

다음의 보조정리들은 정리 1의 증명을 위해 사용된다. 다음 보조정리 1은 재생 부호의 (n,k)-복원 속성, 즉 일반적인 (n,k)-erasure code의 특성으로부터 유도된다.

보조정리 1. 어떤 (n,k,d)-재생 부호가 선형 블록 코드라고 하자. 생성 행렬 G의 $n\alpha$개의 열들을 순서대로 α개씩 묶어서 만든 n개의 $B \times \alpha$ 크기의 부분행렬들을 $G_1, ..., G_n$이라고 하자. 이 n개의 부분행렬들 중 임의의 k개의 행렬을 모아 만든 $B \times k\alpha$ 크기의 행렬은 항상 full rank이다.

증명. $\bar{c} = \overline{m} G$를 만족하는 임의의 $1 \times n\alpha$ 코드 벡터 $\bar{c}$를 고려하자. $\bar{c}$의 심볼들을 α개씩 n의 부분벡터로 나눠어 $\bar{c} = (\overline{c_1}, \overline{c_2}, ... \overline{c_n})$로 나타내었을 때, 재생 부호의 (n,k)-복원 속성에 의해, 임의의 k $(< n)$개의 부분 벡터들로부터 메시지 벡터 $\overline{m}$을 복구할 수 있어야 한다. 이는 해당 k개의 부분벡터들의 인덱스에 대응되는 G의 k개의 부분 행렬들을 모아 만든 $B \times k\alpha$크기의 행렬이 full rank임을 의미한다.■

보조정리 2. 임의의 직교 행렬 Q가 있을 때 (즉, $Q^T Q = QQ^T = I$), Q를 다음과 같이 4개의 부분 행렬로 나누었다고 가정하자.

$$Q = \begin{bmatrix} Q_1 & Q_2 \\ Q_3 & Q_4 \end{bmatrix}$$

이 때, Q_1이 full rank이면, Q_4도 full rank이며, 그 반대도 성립한다.

증명. Q의 크기를 $n \times n$, Q_1의 크기를 $a_1 \times b_1$, Q_4의 크기를 $a_2 \times b_2$라고 가정하자. 즉 $a_1 + a_2 = b_1 + b_2 = n$이다. Q_1과 Q_4 중 행의 수가 열의 수와 같거나 더 많은 행렬이 반드시 존재하며, 일단 Q_4가 그러하다고 가정하자($a_2 \geq b_2$). Q_4가 full rank가 아니라면, Q_4의 b_2개의 열들은 서로 선형 독립 관계에 있지 않으며, $Q_4 \bar{v} = 0$을 만족하는 $a_2 \times 1$ 벡터 $\bar{v}$가 존재한다. $\begin{bmatrix} Q_2 \\ Q_4 \end{bmatrix} \bar{v} = \begin{bmatrix} \overline{w} \\ 0 \end{bmatrix}$이라고 가정하자. 여기서, 만약 $\overline{w} = 0$이라면, $\begin{bmatrix} Q_2 \\ Q_4 \end{bmatrix}$의 모든 열들이 직교하다는 성질을 위배하므로, $\overline{w} \neq 0$이다.

직교 행렬의 성질에 의해, $\begin{bmatrix} \overline{w} \\ 0 \end{bmatrix}$는 $\begin{bmatrix} Q_1 \\ Q_3 \end{bmatrix}$의 모든 열들과 직교하다. 즉 $\begin{bmatrix} \overline{w} \\ 0 \end{bmatrix}^T \begin{bmatrix} Q_1 \\ Q_3 \end{bmatrix} = 0$이고, $\overline{w}^T Q_1 = 0$을 만족한다. Q_1은 행의 수가 열의 수보다 작거나 같으므로, $\overline{w}^T Q_1 = 0$는 $\overline{w}$가 $S(Q_1^T)$에 속하지 않음을 의미한다. 즉 Q_1은 full rank가 아니다.

이번엔 반대로, Q_1이 full rank가 아니라고 가정하자. 이 때,

$$Q^T = \begin{bmatrix} Q_1^T & Q_3^T \\ Q_2^T & Q_4^T \end{bmatrix}$$

도 마찬가지로 직교 행렬이며, Q_1^T이 행의 수가 열의 수와 같거나 많은 행렬에 해당하므로, 위와 같은 방법으로 Q_1^T가 full rank가 아닌 경우, Q_4^T도 full rank가 아님을 증명할 수 있다.■

2.2 재생 부호의 패리티 체크 행렬 H의 조건

다음의 정리 2는 어떤 $(n\alpha, B)$-선형 블록 코드가 (n,k,d)-재생 부호라면, 해당 코드의 패리티 체크 행렬 H가 만족해야 하는 조건을 제시한다.

정리 2. 어떤 (n,k,d)-재생 부호가 선형 블록 코드라면, 그의 패리티 체크 행렬 H는 다음의 두 조건 (i), (ii)를 만족한다.

(i) H의 $n\alpha$개의 열들을 순서대로 α개씩 묶어서 만든 n개의 $(n\alpha - B) \times \alpha$ 크기의 부분행렬

들을 $H_1,\ldots,H_n$이라고 하자. 이 n개의 부분행렬들 중 임의의 $n-k$개의 행렬을 모아 만든 $(n\alpha-B)\times(n-k)\alpha$ 크기의 행렬은 항상 full rank이다.

(ii) $|D|=d$, $i\notin D$를 만족하는 임의의 $i\in[n]$, $D\subset[n]$에 대해, 다음의 조건 (a), (b)를 만족하는 $\alpha\times n\alpha$크기의 행렬 $H_{i,D}$가 항상 존재한다.

(a) $H_{i,D}$를 이루는 n개의 $\alpha\times\alpha$ 행렬들을 각각 $H^1_{i,D},\ldots,H^n_{i,D}$이라 하였을 때 (즉 $H_{i,D}=[H^1_{i,D},H^2_{i,D},\ldots,H^n_{i,D}]$), 다음이 성립한다.

$$\begin{cases} H^j_{i,D}=I_{\alpha,} & \text{if } j=i,\\ rank(H^j_{i,D})\leqq\beta & \text{if } j\in D,\\ H^j_{i,D}=0 & otherwise. \end{cases} \tag{3}$$

(b) $S(H^T_{i,D})\subset S(H^T)$.

증명. 선형 블록 코드의 특성에 따라, 생성 행렬 G와 패리티 체크 행렬 H의 행공간들은 서로 orthogonal complement 관계를 가진다. 즉 G와 같은 행공간을 갖는 행렬 G'과 H와 같은 행공간을 갖는 행렬 H'가 있을 때,

$$[G'^{T}H'^{T}]\begin{bmatrix}G'\\H'\end{bmatrix}=0$$

을 만족하도록 G'과 H'을 잡을 수 있으며, 이는 $\begin{bmatrix}G'\\H'\end{bmatrix}$이 직교 행렬임을 의미한다.

$K\subset[n]$, $|K|=k$을 만족하는 임의의 집합 K를 가정하고, $K'=[n]\setminus K$라 하자. G'의 n개의 $B\times\alpha$크기의 부분 행렬 중 K에 해당하는 k개의 행렬들만을 모아 만든 행렬을 G'_K라고 하고, K'에 해당하는 $n-k$개의 행렬들만을 모아 만든 행렬을 $G'_{K'}$이라 하자. 비슷한 방법으로, H'_K, $H'_{K'}$을 정의할 수 있다. 보조 정리 1에 의해 G_K가 full rank임을 알수 있고, G'_K도 G_K와 같은 행공간을 가지므로, full rank임을 알 수 있다. 또한 보조정리 2에 의해 G'_K가 full rank이면 $H'_{K'}$이 full rank임을 알 수 있으며, $H_{K'}$역시 $H'_{K'}$과 같은 행공간을 가지므로 full rank이다. 이는 (i)이 성립함을 의미한다.

조건 (ii)는 재생 부호의 노드 복구 속성에서 기원한다. $D=\{l_1,\ldots,l_d\}(D\subset[n])$의 인덱스를 갖는 d개의 도움 노드들을 이용하여 $i\notin D$번째 노드를 복구하는 과정을 생각해 보자. l_j번째 도움 노드는 $\overline{c_j}$에 해당하는 심볼들을 가지고 있으며, 이들을 이용하여 β개의 심볼들을 생성하여 i번째 노드로 전송한다. $j\in D$번째 도움 노드로부터 i번째 노드로 전송되는 β개의 심볼들을 $\overline{c_j}\Phi_{ji}$라고 하자. 여기서 Φ_{ji}는 $\alpha\times\beta$ 행렬이다. i번째 노드는 $\overline{c_{l_1}}\Phi_{l_1i}$, $\overline{c_{l_2}}\Phi_{l_2i}$,..., $\overline{c_{l_d}}\Phi_{l_di}$에 해당하는 $d\beta$개의 심볼들을 d개의 도움 노드들로부터 전송받으며, 이를 이용하여 $\overline{c_i}$를 다음과 같이 복구할 수 있다.

$$\overline{c_i}=\sum_{j=1}^{d}(\overline{c_{l_j}}\Phi_{l_ji})\Psi_{l_ji} \tag{4}$$

여기서, $\Psi_{l_1i},\Psi_{l_2i},\ldots,\Psi_{l_di}$는 $\beta\times\alpha$ 행렬이다. $H_{i,D}=[H^1_{i,D},H^2_{i,D},\ldots,H^n_{i,D}]$를 다음과 같이 정의하자.

$$H^j_{i,D}=\begin{cases} I_{\alpha,} & \text{if } t=i,\\ -\Psi^T_{ji}\Phi^T_{ij}, & \text{if } t\in D,\\ 0, & otherwise. \end{cases} \tag{5}$$

$rank(\Phi_{ij}\Psi_{ji})\leqq\beta$이므로, (5)와 같이 정의된 $H_{i,D}$는 (4)를 만족하므로, 조건 (ii)-(a)는 자동으로 만족된다. 또한 임의의 코드워드 벡터 $\overline{c}$에 대해,

$$\begin{aligned} \overline{c}H^T_{i,D} &= \sum_{j=1}^{n}\overline{c_j}(H^j_{i,D})^T\\ &= \overline{c_i}I_\alpha+\sum_{j=1}^{d}\overline{c_{l_j}}(-\Psi^T_{l_ji}\Phi^T_{il_j})^T\\ &= \overline{c_i}-\sum_{j=1}^{d}\overline{c_{l_j}}\Phi_{il_j}\Psi_{l_ji}\\ &= 0. \end{aligned} \tag{6}$$

이 만족되므로, $H_{i,D}$의 모든 행들은 G의 모든 행들에 수직이며, 이는 $S(H^T_{i,D})\subset S(H^T)$임을 의미

한다.■

다음의 정리 3은 정리 2의 조건 (i)의 변형이다.

정리 3. 임의의 (n,k,d)-재생 부호의 패리티 체크 행렬 H가 있을 때, $K\subset[n]$, $|K|=k$을 만족하는 임의의 집합 K에 대해, 다음의 조건들을 만족하는 $(n-k)\alpha\times n\alpha$크기의 행렬 $M_K=\begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n-k,1} & A_{n-k,2} & \cdots & A_{n-k,n} \end{bmatrix}$가 존재한다. 이 때, $A_{i,j}$ $(i\in[n-k], j\in[n])$는 M_K의 $\alpha\times\alpha$ 크기의 부분 행렬들을 의미한다.

(i) $K'=[n]\setminus K=\{l_1,\ldots,l_{n-k}\}$일 때, $\begin{bmatrix} A_{1,l_1} & A_{1,l_2} & \cdots & A_{1,l_{n-k}} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n-k,l_1} & A_{n-k,l_2} & \cdots & A_{n-k,l_{n-k}} \end{bmatrix}=I_{(n-k)\alpha}$.

(ii) $S(M_K^T)\subset S(H^T)$

증명. H의 n개의 $(n\alpha-B)\times\alpha$크기의 부분행렬들 중 K'에 해당하는 것들만을 모아 만든 $(n\alpha-B)\times(n-k)\alpha$ 크기의 행렬을 $H_{K'}$이라 하자. $H_{K'}$은 full rank이며, $(n-k)\alpha\leqq(n\alpha-B)$이므로, $S(H_{K'}^T)=\mathbb{R}^{(n-k)\alpha}$이다. 즉, $MH_{K'}=I_{(n-k)\alpha}$를 만족하는 $(n-k)\alpha\times(n\alpha-B)$크기의 행렬 M이 존재한다. $M_{K'}=MH$로 정의하면, $M_{K'}$는 위의 두 조건 (i), (ii)를 만족한다.■

2.3 정리 1의 증명

이번 장에서는 본 논문의 주된 결과인 정리 1의 증명에 대해 다룬다. (n,k,d)-재생 부호가 있을 때, 일반성을 잃지 않고, $k\leqq\bar{d}\leqq d$인 정수 $\bar{d}$에 대하여, $1,2,\ldots,\bar{d}$의 노드를 이용하여 $d+1$번째 노드를 복구하는 상황을 가정하자. $D=[d]$, $i=d+1$에 대해 정리 2의 조건 (ii)에 의해 제시되는 $\alpha\times n\alpha$ 행렬 $H_{d+1,[d]}$를 생각해 보자. 이 때 $H_{n,[d]}$는 정리 2의 조건 (ii)-(a), (ii)-(b)를 만족한다. $\tilde{\beta}$를 다음과 같이 정의하자.

$$\tilde{\beta}\equiv rank([H_{d+1,[d]}^{\bar{d}+1}\,H_{d+1,[d]}^{\bar{d}+2}\cdots H_{d+1,[d]}^{d}]) \quad (7)$$

여기서, $\tilde{\beta}$가 $\binom{\bar{d}}{k}$의 배수인 경우를 가정하자($\tilde{\beta}$ 및 α,β등의 크기는 B의 크기를 바꿈으로써 조절 가능하다.). $\delta=\tilde{\beta}/\binom{\bar{d}}{k}$일 때, $[H_{d+1,[d]}^{\bar{d}+1}\,H_{d+1,[d]}^{\bar{d}+2}\cdots H_{d+1,[d]}^{d}]$는 각각 $rank$가 δ인 $\binom{\bar{d}}{k}$개의 행렬들의 합으로 나타낼 수 있다. 그 각각의 $\alpha\times(d-\bar{d})\alpha$ 크기의 행렬들을 $\Delta_{K_1},\Delta_{K_2},\ldots,\Delta_{K_{\binom{\bar{d}}{k}}}$라고 하자. 즉,

$$[H_{d+1,[d]}^{\bar{d}+1}\,H_{d+1,[d]}^{\bar{d}+2}\cdots H_{d+1,[d]}^{d}]=\sum_{i=1}^{\binom{\bar{d}}{k}}\Delta_{K_i} \quad (8)$$

이 된다. 여기서, $K_1,\ldots,K_{\binom{\bar{d}}{k}}$는 각각 원소의 개수가 k개인 $[\bar{d}]$의 부분집합들을 의미한다.

$1\leqq l\leqq\binom{\bar{d}}{k}$를 만족하는 임의의 l을 선택하자. l과 대응되는 K_l에 대해, 정리 3에 의해 제시되는 $k\alpha\times n\alpha$ 크기의 행렬 M_{K_l}가 존재한다. $K_l\subset[\bar{d}]$이므로, $K'_l=[n]\setminus K_l$은 $[\bar{d}+1,d]$를 포함한다. 다음의 식으로 정의되는 $\alpha\times n\alpha$ 행렬 $\widetilde{H}$를 정의하자.

$$\widetilde{H}=H_{d+1,[d]}-\sum_{l=1}^{\binom{\bar{d}}{k}}[0_{\alpha\times(n-k-d+\bar{d})\alpha}\Delta_{K_l}]M_{K_l} \quad (9)$$

여기서, $\widetilde{H}$의 $\alpha\times\alpha$크기의 n개의 부분행렬들을 각각 $\widetilde{H}_1,\ldots,\widetilde{H}_n$이라고 하면(즉 $\widetilde{H}=[\widetilde{H}_1\cdots\widetilde{H}_n]$), 다음이 성립한다.

$$\begin{cases}\widetilde{H}_j=I_{\alpha,} & \text{if } j=d+1,\\ rank(\widetilde{H}_j)\leqq\beta+\binom{\bar{d}}{k}k\delta/\bar{d} & \text{if } j\in[\bar{d}],\\ \widetilde{H}_j=0 & otherwise.\end{cases} \quad (10)$$

여기서,

$$\begin{aligned}\beta+\binom{\bar{d}}{k}k\delta/\bar{d} &= \beta+\frac{k}{\bar{d}}\tilde{\beta}\\ &\leqq \beta+\frac{k(d-\bar{d})\beta}{\bar{d}}\\ &= \frac{d\beta+(k-1)(d-\bar{d})\beta}{\bar{d}}\\ &= \bar{\gamma}/\bar{d}\end{aligned} \tag{11}$$

이다. $\widetilde{H}$의 모든 행들은 $H_{d+1,[d]}$와 M_{K_l}의 행들의 선형 조합이므로, $S(\widetilde{H}^T)\subset S(H^T)$를 만족한다. $\bar{c}=\overline{m}G$에 의해 생성된 코드워드 벡터들은 모두 $S(H^T)$에 직교하므로, $\bar{c}^T\widetilde{H}=0$을 만족한다.

$$\begin{aligned}\bar{c}_{d+1}^T &= \bar{c}_{d+1}^T\widetilde{H}_{d+1}\\ &= \bar{c}_{d+1}^T\widetilde{H}_{d+1}-\bar{c}^T\widetilde{H}\\ &= -\sum_{\substack{t=1\\ t\neq d+1}}^{n}\bar{c}_t^T\widetilde{H}_t\\ &= -\sum_{t=1}^{\bar{d}}\bar{c}_t^T\widetilde{H}_t\end{aligned} \tag{12}$$

즉, $d+1$번째 노드의 정보인 $\bar{c}_{d+1}$은 $\bar{d}$개의 노드들의 심볼들($\overline{c_1},\ldots,\overline{c_{\bar{d}}}$)의 선형 조합으로 나타낼 수 있다. 또한, $rank(H_t)\leqq\bar{\gamma}/\bar{d}$이고, 이는 각 $\bar{d}$개의 노드들에서 각각 $\bar{\gamma}/\bar{d}$개씩의 심볼들의 다운로드만이 필요함을 의미한다. 이는 정리 1의 증명이 완료되었음을 의미한다.

Ⅲ. 결 론

(n,k,d)-재생 부호는 분산 저장 시스템에 적용되었을 때, erasure code가 가지는 저장 용량 효율성을 가지며 동시에 d개의 살아남은 노드들의 도움을 통해 한 개의 손실된 노드를 낮은 복구 대역폭으로 복구할 수 있는 코드이다. 하지만, 기존의 재생 부호는 노드 손실이 빈번하거나, 노드 간 접속이 불안정한 환경에서, d개 이하의 노드들에만 접속 가능할 경우에 유연하게 대처할 수 없다는 단점이 있다. 본 논문에서는 약간의 복구 대역폭의 희생을 통하여, $k\leqq\bar{d}\leqq d$의 다양한 도움 노드의 수 $\bar{d}$개로 노드를 복구할 수 있는 유연한 코드 운용 방식을 제안하였다. 해당 기법은 임의의 (n,k,d)-재생 부호들에 모두 적용 가능하며, 선형 블록 코드의 특성들을 활용하여, d개 이하의 도움 노드를 사용할 경우 기존의 $k\alpha$보다 훨씬 작은 $\bar{\gamma}=d\beta+(k-1)(d-\bar{d})\beta$의 복구 대역폭을 가짐을 증명하였다.

References

[1] D. Borthakur, *HDFS architectuure guide, Hadoop apache project*(2008), http://hadoop.apache.org/common/docs/current/hdfs_design.pdf

[2] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. F. ul Haq, M. I. ul Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, and L. Rigas, "Windows azure storage: A highly available cloud storage service with strong consistency," in *Proc. ACM SOSP*, pp. 143-157, Oct. 2011.

[3] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," in *Proc. USENIX Workshop on Hot Topics in Storage and File Systems*, Jun. 2013.

[4] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843-5855, Oct. 2014.

[5] J.-H. Kim, M.-Y. Nam, and H.-Y. Song, "Construction of [2^k-1+k, k, 2^k-1+1] codes attaining Griesmer bound and its locality," *J. KICS*, vol. 40, no. 03, Mar. 2015.

[6] M.-Y. Nam, J.-H. Kim, and H.-Y. Song, "Locally repairable fractional repetition codes," *J. KICS*, vol. 40, no. 9, pp. 1741-1753. Sept. 2015.

[7] J.-H. Kim, M.-Y. Nam, and H.-Y. Song, "Bianry locally repairable codes from

complete multipartite graphs," *J. KICS*, vol. 40, no. 03, Mar. 2015.

[8] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539-4551, Sept. 2010.

[9] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227-5239, Aug. 2011.

이 혁 (Hyuk Lee)

2011년 2월 : 포항공과대학교 전기전자공학과 학사

2013년 2월 : 서울대학교 전기·정보공학부 석사

2013년 3월~현재 : 서울대학교 전기·정보공학부 박사과정

<관심분야> 무선통신, 분산저장시스템

이 정 우 (Jungwoo Lee)

1988년 : 서울대학교 전기공학 학사

1990년 : Princeton대학교 전기공학 석사

1994년 : Princeton대학교 전기공학 박사

2002년~현재 : 서울대학교 전기·정보공학부 교수

<관심분야> 무선통신, 분산저장시스템 부호, 머신러닝