분산 저장 시스템을 위한 부분접속 복구부호

남미영, 김정현, 송홍엽
연세대학교

요 약

본고에서는 분산 저장 시스템의 성능 척도를 규정하고 그 중, 복구 과정에서의 부분접속수를 향상시킬 수 있는 부호화 기법에 대해 살펴본다.

Ⅰ. 서 론

클라우드 컴퓨팅은 정보기술 분야에서의 화두로 떠오르고 있다. 클라우드 컴퓨팅은 네트워크 서버, 스토리지, 애플리케이션, 서비스 등의 IT 자원을 개별적으로 소유하지 않고 필요할 때마다 인터넷을 통해 편하고 빠르게 서비스 받는 방식이다[2]. 클라우드 컴퓨팅을 통한 서비스는 이미 많은 사람들이 친숙하게 사용하고 있다. 해외의 대표적인 클라우드 서비스로는 Amazon의 EC2(Elastic Cloud Computing)와 S3(Simple Storage Service), Google의 AppEngine, 그리고 Microsoft의 Azure등을 꼽을 수 있다. 국내에서는 다양한 클라우드 서비스 중에서 클라우드 스토리지 서비스가 가장 널리 사용되고 있다. 국내 서비스로는 네이버의 N 드라이브, 다음의 다음 클라우드 등이 대표적이다.

클라우드 컴퓨팅 시장에 뛰어든 기업들에게 막대한 양의 데이터를 저장하고 관리하기 위한 비용을 줄이는 것은 중요한 이슈이다. 빈번한 장애로부터 데이터의 영구적인 소실을 막기 위한 안정적인 데이터 저장 방법이 마련되어야 한다. 가장 간단하게 데이터를 중복 저장하는 방법에서부터 좀 더 복잡하지만 효율적인 소실 부호(erasure coding)를 적용하는 기법 등이 존재한다[4][5].

본고에서는 이러한 분산 저장 시스템에 부합할 수 있는 부호화 기법들을 소개한다. 본고의 구성은 다음과 같다. Ⅱ장에서는 분산 저장 시스템에 대해 간략히 살펴보고 Ⅲ장에서는 분산저장 시스템에서의 성능 척도에 대해 다룬다. Ⅳ장에서는 성능 척도 중 하나인 복구 과정에서의 부분접속수 성능을 향상시킬 수 있는 부분접속 복구 부호의 구체적인 예를 다룬다. 마지막으로 Ⅴ장에서는 결론을 내린다.

Ⅱ. 분산 저장 시스템

1. 분산 저장 시스템의 기능

분산 저장 시스템을 위해 사용되는 분산 파일 시스템(Distributed File System)은 데이터의 저장과 관리뿐만 아니라 상위 계층 서비스가 요구하는 충분한 성능과 안정성을 보장해준다. 수많은 분산 파일 시스템 중 실제 클라우드 컴퓨팅에 활용되고 있는 분산 파일 시스템은 Google의 파일 시스템(GFS)과 Apache의 Hadoop 분산 파일 시스템 (HDFS)[3]이다. 본고에서는 Amazon, IBM, Facebook 등에서 사용되고 있는 HDFS를 기본 파일시스템으로 가정하고 설명하겠다.

분산 저장 시스템은 크게 데이터를 저장, 복구, 사용하는 3가지의 동작으로 그 기능을 분류할 수 있다.

사용자가 크기 M인 파일을 분산 저장 시스템에 저장하고자 하는 경우, 파일을 k개의 데이터 블록으로 나누어 $n-k$개의 redundancy를 붙여 n개의 데이터노드에 분산 저장한다. HDFS의 기본 동작인 3회 반복 부호를 사용하는 경우 한 개의 블록마다 두 개의 사본을 만들어 총 3개의 블록을 세 노드에 나누어 저장한다. 이 과정이 〈그림 1〉의 저장 과정에 해당한다.

사용자가 저장된 데이터를 사용하기 위해서는 자신의 데이터 블록을 저장하고 있는 K개의 데이터 노드로부터 저장된 블록을 받아와 소스 파일을 생성한다. 이는 〈그림 1〉에서 사용의 과정에 해당된다.

분산 저장 시스템을 구성하는 데이터 노드들은 각각이 불완전한 특성을 갖기 때문에 노드에 장애가 발생하거나 네트워크에서 접속이 끊기는 등의 상황이 발생하여 사용할 수 없는 상태가 되기도 한다. 이러한 경우에도 시스템의 안정성을 일정하게 유지하기 위해 복구(repair) 과정이 필요하다. 복구 과정은 n개의

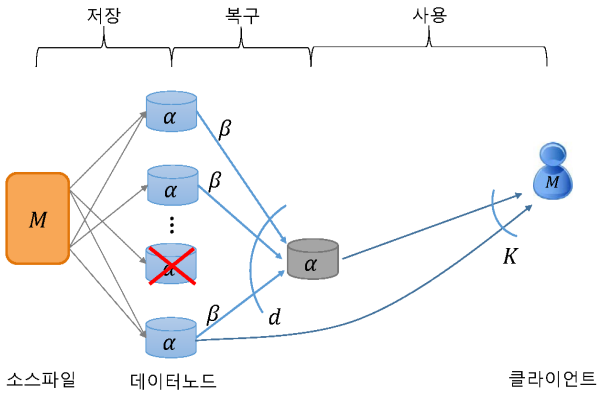

그림 1. 분산 저장 시스템의 기능

데이터 노드를 주기적으로 감시하여 장애가 발생한 노드가 발견되면 해당 노드에 저장된 데이터 블록을 재생하여 새로운 노드에 저장하는 과정이다. 이때 소실된 데이터 블록의 재생은 나머지 $n-1$개의 노드들 중 일부 d개의 노드로부터 저장된 데이터 블록을 받아옴으로써 이루어진다. 이 과정이 〈그림 1〉의 복구에 해당한다.

2. 전통적인 부호의 적용

HDFS의 기본 동작인 3회 반복 기법은 간단하게 안정성을 향상시킬 수 있는 방법이지만 시스템에 저장해야 할 블록의 양이 크게 늘어난다는 단점을 갖는다. 3회 반복의 경우 200%의 오버헤드가 발생한다. 데이터의 양이 늘어남에 따라 실제 분산파일시스템이 저장하고 관리해야 하는 데이터의 증가량은 훨씬 커져, 시스템 용량을 벗어나게 된다. 이를 해결하기 위해서단순 반복 기법보다는 좀 더 효율적인 부호화 기법의 적용을 모색하고 있다.

Facebook에서는 Reed-Solomon(RS) 부호를 사용하여 데이터에 대한 redundancy를 생성함으로써 데이터의 안정성을 높여주는 기법을 사용한다. RS 부호에 기반한 HDFS-RAID라는 오픈 소스 모듈을 적용하였다.

RS부호는 최대 거리 분리 가능(Maximum Distance Separable; MDS) 부호로서, 동일한 안정성을 갖는 부호 중 최소의 저장 공간을 사용하는 부호이다. 따라서 저장 공간 오버헤드 측면에서 최적이다.

HDFS-RAID에서는 10개의 데이터 블록이 포함된 스트라이프(stripe)를 하나의 부호화 단위로 사용한다. 즉 서로 다른 스트라이프마다 (14,10) RS 부호를 이용해 부호화 한다. 10개의 데이터블록을 14개의 블록으로 부호화하기 때문에 저장 공간 오버헤드는 40%이다. 반복 부호에서의 200% 오버헤드에 비해 사용되는 저장 공간을 획기적으로 줄일 수 있다.

〈그림 2〉에서는 반복 부호와 MDS 부호의 복구 과정에서의 효율성을 비교하였다. 반복 부호를 사용한 경우

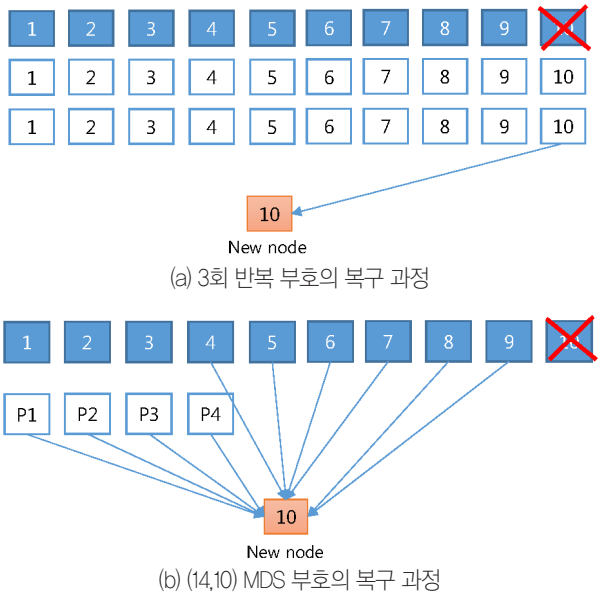

그림 2. 반복부호와 MDS 부호의 복구 과정

문제가 발생한 노드에 저장돼 있던 데이터를 사본을 저장하고 있는 다른 노드로부터 그대로 복사함으로써 복구가 가능하다. 반면 MDS 부호를 사용한 경우는 (b)처럼 k개의 데이터를 받아오고 이를 이용한 복호화 과정을 통해, 원래의 파일을 만들어내어 그로부터 소실된 데이터를 재생한다. 빈번한 노드의 장애가 발생하는 시스템에서 한 개의 데이터를 재생하기 위해 k개의 데이터 전송이 필요하기 때문에 이 과정이 전체 시스템의 병목이 될 수 있다. 이러한 문제점으로 인해 Facebook에서도 전체 데이터의 8%만 RS 부호화를 통해 저장한다. 그럼에도 불구하고 복구 과정을 위한 데이터 전송량은 전체 전송량의 30%에 육박한다[10].

기존의 통신 시스템에 사용되는 오류정정부호는 복구의 과정을 전혀 고려하지 않았다. 전체 데이터의 효율적인 복호를 목표로 하였던 기존의 오류정정부호가 분산 저장 시스템에서 새로이 요구되는 기능인 일부 데이터의 복구는 효과적으로 수행하지 못한다. 따라서 새로운 시스템에 적합한 새로운 부호화 기법의 도입이 필요하다.

Ⅲ. 분산 저장 시스템의 성능 척도

1. 데이터노드의 저장용량

분산 저장 시스템에 n개의 노드가 존재하고 각 노드에 요구되는 저장용량은 〈그림 1〉의 α에 해당하는 값이다. 시스템에 저

장해야 하는 데이터의 양이 급격히 증가하는 추세에 따라 이 값을 줄이는 것이 중요하다.

2. 복구 대역폭

노드에 장애가 발생하였을 때 복구과정을 수행해야 한다. 이 때 남아 있는 노드들로부터 저장된 데이터를 받아와 해당 노드가 저장하고 있던 데이터를 복구해야 한다. 이 과정에서 발생하는 총 데이터 전송량을 복구 대역폭(repair bandwidth)이라고 한다[6]. 〈그림 1〉에서는 d개의 노드에 접속해 각 노드로부터 β 만큼의 데이터를 받아와 복구하고 있다. 복구 대역폭은 일반적으로 $\gamma = d\beta$로 표기한다. 분산 저장 시스템에서는 복구를 위한 네트워크 사용량이 전체 네트워크 사용량에서 큰 비중을 차지한다. 따라서 이 값을 최소화 하는 것이 매우 중요하다.

3. 부분접속수

일부 시스템에서는 하나의 노드를 복구하기 위해 다운로드하는 총 데이터 양 보다 접속해야 하는 노드의 수가 더 중요한 지표가 된다. 이는 〈그림 1〉의 d에 해당하는 값으로, 부분접속수(locality)라고 한다[7]. 실제로 많은 사용자들이 수신하고자 하는 인기 있는 데이터(hot data)를 저장하고 있는 노드의 경우, 클라이언트로부터 사용을 위한 접속 요청이 많아 이를 처리하기 위해 다른 요청에 응답하지 못하는 경우가 생긴다. 따라서 빈번하게 발생하는 복구 과정에서 한 노드의 복구를 위해 접속해야 하는 노드의 수를 줄이는 것이 시스템의 원활한 동작을 돕는다.

부분접속수를 향상시킬 수 있는 부호에 대한 연구가 활발히 진행되고 있다. 특히 부호의 부분접속수와 최소거리 사이의 관계에 대한 연구[7]를 시작으로, 주어진 부분접속수를 갖고 동시에 최적의 최소 거리를 달성할 수 있는 부호를 설계하는 연구들이 활발히 진행되고 있다[4][12][13][14]. 다음 장에서는 낮은 부분접속수를 갖는 부호인 부분접속 복구 부호에 대해서 살펴본다.

Ⅳ. 부분접속 복구 부호

본 장에서는 분산 저장 시스템에 적합한 부호의 하나인 부분접속 복구 부호(Locally Repairable Code; LRC)에 대해서 다룬다.

낮은 부분접속수를 갖는 부호를 통틀어 부분접속 복구 부호라고 한다. 부분접속수가 $r \ll k$인 $[n,k]$부호의 최소거리의 바운드는 다음과 같다[7].

$$d_{\min} \le n - k + 1 - \left(\left\lceil \frac{k}{r} \right\rceil - 1 \right)$$

이 식을 통해, 부분접속수 r의 값이 작아질수록 부호의 최소거리가 가질 수 있는 최댓값이 작아짐을 알 수 있다. 즉, 작은 부분접속수를 갖기 위해서 부호의 최소거리에서 손해를 보아야 하는 상관관계가 있음을 알 수 있다. 최소거리가 싱글톤(Singleton) 바운드를 달성하는 MDS 부호는 부분접속수가 $r = k$로, 가장 큰 부분접속수를 갖는 부호이다. MDS 부호를 사용할 경우 한 노드를 복구하기 위해서 k개의 다른 노드에 접속해야 하는 것이다.

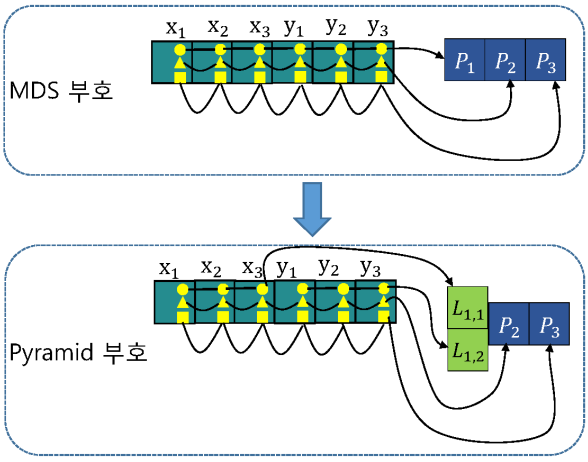

그림 3. Pyramid 부호의 생성

1. Windows Azure에 사용되는 부분접속 복구 부호

Microsoft사의 Azure에 사용되는 LRC는 기본적으로 Pyramid code[8]와 유사한 형태이다[9]. 본 장에서는 편의를 위해 실제 시스템에 적용된 부호보다 작은 크기를 갖는 LRC를 예로 들어 설명하겠다.

MDS 부호인 RS 부호를 이용해 LRC를 생성한다. $(9,6)$ RS 부호는 6개의 데이터 심볼에 3개의 패리티 심볼을 추가하는 부호화 과정을 거쳐 9개의 심볼을 생성한다. 앞서 설명한대로 RS 부호에서 한 심볼만 복구하는 과정에도 6개의 심볼이 모두 필요하다. 기존의 RS 부호의 3개의 패리티 심볼은 모두 6개의 데이터 심볼의 선형조합을 통해 계산된다. 이와는 달리, 여섯 개의 데이터 심볼을 세 개씩 두 그룹으로 나누고, 각 그룹에 패리티 심볼을 하나씩 생성한다. 이를 지역(local) 패리티라고 한다. 〈그림 3〉은 이 과정을 나타낸다.

첫 번째 지역 패리티 $L_{1,1}$은 첫 번째 그룹에 속하는 x_1, x_2, x_3 3개의 데이터 심볼의 선형연산으로 구한 패리티이다. $L_{1,2}$는 두 번째 그룹인 y_1, y_2, y_3의 선형연산 결과이다. 데이터 심볼과 지역 패리티 심볼 중 한 개의 심볼만 소실된 경우, 이 지역패리티 연산에 관여하는 나머지 세 심볼로부터 소실된 심볼을 복구할 수 있다. 예를 들어 x_1이 소실된 경우 $L_{1,1} = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$를 이용해,

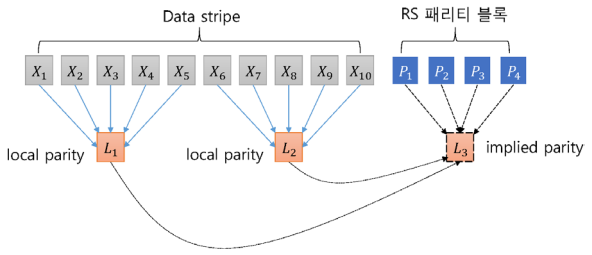

그림 4. Facebook의 LRC

$x_1 = \lambda_1^{-1}(L_{1,1} - \lambda_2 x_2 - \lambda_3 x_3)$로 복구가 가능하다. 글로벌 패리티 P_2와 P_3를 제외한 나머지 모든 심볼에 대해서 이와 유사한 부분접속 복구가 가능하다. 이러한 부분접속 복구 부호는 information symbol locality를 갖는다고 한다.

2. Facebook에 사용되는 부분접속 복구 부호

Facebook에서는 기존의 3회 반복 부호 대신 RS 부호에 기반한 HDFS-RAID를 만들었지만 이 역시 부분접속수와 복구대역폭의 비효율성의 문제가 존재한다. 이를 개선하기 위해 효율적인 복구가 가능한 HDFS-Xorbas를 고안하였다[10].

먼저 10개의 데이터 블록에 (14,10) RS 부호를 적용해 14개의 부호화 블록을 생성한다. 그리고 데이터 블록을 5개씩 두 그룹으로 나누어 각각의 지역 패리티 블록 L_1과 L_2를 생성한다. 나머지 RS 패리티 블록에 해당하는 P_1, P_2, P_3, P_4에 대해서도 지역 패리티 블록 L_3를 생성하는데, $L_3 = \sum_{i=1} P_i + L_1 + L_2$를 만족하도록 한다. 이를 통해 L_3는 실제로 저장하지 않아도 알 수 있는 implied parity가 되어, 저장 공간 오버헤드를 줄일 수 있다. 이렇게 생성된 부호는 모든 심볼이 부분접속수 $r = 5$를 갖는다. 이러한 부분접속 복구 부호는 all symbol locality를 갖는다고 한다.

3. 순환 부분접속 복구 부호

순환부호를 부호의 근(root)를 이용해 정의함으로써 부호의 최소 거리 특성과 부분접속수 특성을 특정 지을 수 있다[11].

유한 체 F_q의 primitive n-th root of unity α를 이용하여, 두 개의 집합 D와 L을 다음과 같이 정의할 수 있다.

$L = \{\alpha^i, i \bmod (r+1) = l\}$, where l, $0 \le l \le r$, be an integer.

$D = \{\alpha^{j+sb}, s = 0, \ldots, n - k/r(r+1)\}$, where $\alpha^j \in L$.

이 두 개의 집합을 근으로 하는 순환부호는 부분접속수 r을 갖는 (n,k) 순환부호가 된다.

〈그림 5〉는 두 집합 D, L과 이를 근으로 갖는 부호의 최소거리 및 부분접속수 특성을 나타낸다. D는 부호의 최소거리를 특정 짓고 L은 부호의 부분접속수를 결정한다.

이러한 방식으로 생성된 부호를 RS-like 부분접속 복구 부호라고 한다. RS-like 부분접속 복구 부호를 유한 체 F_{q^m}에서 생성하고 이것의 subfield subcode의 부분접속수에 대한 분석도 이루어져 있다[11]. RS-like 부분접속 부호를 F_{2^m}에서 생성하여, 그것의 subfield subcode를 생성함으로써 이진(binary) 부분접속 복구 부호를 생성할 수 있다.

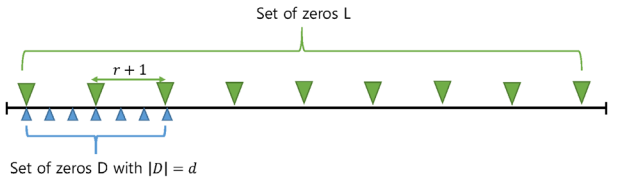

그림 5. 순환 부호의 근의 집합과 최소 거리 및 부분접속수 특성

4. 이진 부분접속 복구 부호

앞에서 살펴본 RS-like부호의 subfield subcode의 생성을 통한 부분접속 복구 부호의 생성은 이진 체(binary field)와 같은 작은 크기의 유한 체에서의 부분접속 복구 부호의 생성을 가능하게 했다. 실제 시스템에서 유한 체의 곱셈 연산은 굉장히 많은 연산량을 요구하고, 이로 인한 전력의 소모나 다른 요청에 대한 응답이 지연되는 등의 문제가 발생하게 된다. 이러한 이유로, 작은 크기의 유한 체에서 부분접속 복구 부호의 생성 방법에 대한 연구가 이루어지고, 그 중 이진 부분접속 복구 부호의 생성방법이 특정 파라미터에서 알려져 있다.

$n = 15, k = 10$을 파라미터로 갖는 이진 부호의 가능한 최소 거리의 최댓값은 $d_{\min} = 4$이다[16]. 〈그림 6〉은 $[n = 15, k = 10, d_{\min} = 4, r = 6]$ 이진 부분접속 복구 부호의 생성행렬이다[15]. 이 부호가 갖는 부분접속수 $r = 6$ 이 주어진 $n, k, d_{\min}$ 파라미터에서 가능한 최소의 부분접속수이다.

$$G = \left[\begin{array}{cccccccccc|cccccccccc} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

P^T $I_{10\times10}$

그림 6. 이진 부분접속 복구 부호의 생성행렬

V. 결 론

본고에서는 분산 저장 시스템에 사용되는 부호화 기법이 시스템의 주요 성능에 큰 영향을 미침을 알아보았다. 특히 복구 과정에서의 부분접속수 성능을 향상시킬 수 있는 부호화 기법의 구체적인 생성 방법을 중점적으로 다루었다. Microsoft나 Facebook 등에서 실제 분산 저장 시스템에 적용하고 있는 간단한 부분접속 복구 부호의 생성방법을 비롯한 더 향상된 다양한 부분접속 복구 부호에 대해서 살펴보았다. 현재까지 잘 알려진 부분접속 복구 부호는 대부분 큰 크기의 유한 체에서 생성된다. 이러한 부호들은 효율성과 성능을 보장함에도 불구하고 시스템의 복잡도를 증가시킨다는 문제점으로 인해 실제 시스템에 적용이 제한적이다. 기존 시스템의 구조를 크게 변형시키지 않으면서 동시에 낮은 복잡도 및 적은 연산 량을 갖는 부분접속 복구 부호에 대한 연구가 활발히 이루어져야 현재 시스템에 직접 적용이 이루어질 것이다.

참 고 문 헌

[1] 김정현, 박진수, 박기현, 남미영, 송홍엽, "차세대 클라우드 저장 시스템을 위한 소실 복구 코딩 기법 동향," 한국통신학회 학회지, 정보와통신, 제 31권, 제 2호, pp. 125-131, 2014년 2월.

[2] 민영수, 진기성, 김홍연, 김영균, "클라우드 컴퓨팅을 위한 분산 파일 시스템 기술 동향," 전자통신동향분석, 제 24권, 제 4호, 2009년 8월.

[3] HDFS, http://hadoop.apache.org/docs/

[4] 김정현, 남미영, 송홍엽, "Griesmer 한계식을 만족하는 [2^k-1+k, k, 2^(k-1)+1] 부호 설계 및 부분접속수 분석", 한국 통신학회 논문지, 2015년 3월 게재예정.

[5] Y. Cassuto, "What can Coding Theory do for Storage Systems?," ACM SIGACT News, vol. 44, no. 1, pp. 80-88, Mar. 2013.

[6] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," IEEE Trans. on Inf. Theory, vol. 56, no. 9, pp. 4539-4551, Sep. 2010.

[7] P. Gopalan, C. Huang, H. Simitci, S. Yekhanin, "On the Locality of Codeword Symbols," IEEE Trans. on Inf. Theory, vol. 58, no. 11, pp. 6925-6934, Nov. 2012.

[8] C. Huang, M. Chen, and J. Li, "Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems," IEEE Int. Symp. Network Computing and Applicaitons, Jul. 2007.

[9] C. Huang, H. Simitci, Y. Xu, X. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure Coding in Windows Azure Storage," USENIX Annual Technical Conference, Jun. 2012.

[10] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing Elephants: Novel Erasure Codes for Big Data," in Proc. VLDB Endowment, pp. 325-336, 2013.

[11] I. Tamo, A. Barg, S. Goparaju, and R. Calderbank, "Cyclic LRC Codes and their Subfield Subcodes," arXiv:1502.01414.

[12] N. Silberstein, A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, "Optimal Locally Repairable Codes via Rank-metric Codes," in Proc. IEEE ISIT 2013, pp. 1819-1823, Jul. 2013.

[13] F. Oggier and A. Datta, "Self-Repairing Codes for Distributed Storage - A Projective Geometric Construction," in Proc. IEEE ITW 2011, pp. 30-34, Oct. 2011.

[14] W. Song, S. H. Dau, C. Yuen, and T. J. Li, "Optimal Locally Repairable Linear Codes," IEEE J-SAC. vol. 32, no. 5, pp. 1019-1036, May 2014.

[15] M. Shahabinejad, M. Khabbazian, and M. Ardakani, "An Efficient Binary Locally Repairable Codes for Hadoop Distributed File System," IEEE Comm. Lett. vol. 18, no. 8, pp. 1287-1290, Jul. 2014.

[16] R. Schürer and W. C. Schmid, "MinT-Architecture and applications of the t, m, s-net and OOA database," Math. Comput. Simul., vol. 80, no. 6, pp. 1124 - 1132, Feb. 2010.

약 력

남 미 영

2005년 연세대학교 전기전자공학과 졸업
2005년~2007년 삼성전자 연구원
2009년 연세대학교 전기전자공학과 석사
2009년~현재 연세대학교 전기전자공학과 박사과정
관심분야: 부호이론, 분산저장시스템

김 정 현

2006년 연세대학교 전기전자공학과 졸업
2008년 연세대학교 전기전자공학과 석사
2010년~2013년 한국전자통신연구원 연구원
2013년~현재 연세대학교 전기전자공학과 박사과정
관심분야: 통신공학, 정보이론, 부호이론, 분산저장시스템

송 홍 엽

1984년 연세대학교 전자 공학과 졸업
1986년 University of Southern California Dept. of EE. Systems 석사
1991년 University of Southern California Dept. of EE. Systems 박사
1992년~1993년 Post-Doc Research Associate, University of Southern California Dept. of EE. Systems
1994년~1995년 Senior Engineer, Qualcomm Inc., San Diego, California.
2002년~2003년 Visiting Professor, University of Waterloo, Canada
1995년~현재 연세대학교 전기전자공학과 교수
관심분야: 통신공학, 정보이론, 부호이론, 암호이론, 이산수학