비디오 스토리지 시스템에서 복구 읽기 I/O 대역폭 최소화를 위한 적응형 중복 방법 선택 기법

Adaptive Redundancy Selection to Minimize I/O Bandwidth for Degraded Read Operations in Video Storage Systems

이춘광, 이다영, 송민석[1)]
Chunguang Li, Dayoung Lee, Minseok Song

(22212) 인천광역시 미추홀구 인하로 100, 인하대학교 컴퓨터공학과
sychunguang@naver.com, dayoung08@inha.edu, mssong@inha.ac.kr

요 약

비디오 서버는 데이터 손실을 방지하기 위해 데이터에 중복 (redundancy) 기법을 적용한다. 중복 방식은 복제 (replication)과 에러복구코드 (erasure code) 2가지가 있다. 복제기법은 필요한 추가 데이터를 위한 저장공간이 크나, 데이터 복구시 읽기 대역폭이 요구량이 작다. 에러복구코드는 추가 데이터 저장 공간을 절약할 수 있지만 읽기 대역폭이 많이 소모한다. 따라서, 2가지 기법을 이용하여 추가 데이터 스토리지 공간 한도 안에서 읽기 대역폭을 줄이는 것은 매우 중요하다. 본 논문에서 파일의 인기도와 크기를 기반으로 필요한 추가 데이터 저장 공간, 읽기 대역폭을 고려해서 파일별 중복 옵션을 선택하는 알고리즘을 제안한다. 이를 위해서 중복 옵션 선택 문제를 공식화하고, 추가 데이터 저장 공간, I/O 대역폭, 비디오 접근률 분포, 비디오의 크기 및 적용하는 RS 코딩 옵션의 측면에서 제안한 알고리즘과 다른 2가지 알고리즘의 결과를 비교하였다.

Abstract

Video servers use data redundancy techniques to prevent data loss. The redundancy technique includes replication and erasure code. Replication requires a large amount of data storage space but the read bandwidth for data recovery is low. However, the erasure code scheme reduces extra data storage space but consumes a large amount of read bandwidth. It is important to combine these two methods effectively to reduce read bandwidth within the limitation of additional data storage space. This paper presents an algorithm to select the redundancy option for each file, considering the tradeoff between additional data storage space and read bandwidth required, based on the popularity and size of the file. To this end, we formulate the redundancy option selection problem, and compare the results of the proposed algorithm with those of other two algorithms in

※ 본 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2021R1F1A1064053).
본 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2022R1A2C1007237).
1) 교신저자

논문투고일자 : 2022. 03. 23 심사일자 : 2022. 03. 29 게재확정일자 : 2022. 04. 18 http://doi.org/10.23019/kingpc.18.2.202204.003

terms of additional data storage space, I/O bandwidth, video access rate distribution, video size and RS coding options.

키워드: 비디오 서버, 스토리지, 데이터 중복, I/O 대역폭
Keyword: Video server, storage, data redundancy, I/O 대역폭

1. 서론

최근 5G기술이 대중화되면서 네트워크 환경 더욱 좋아져서 비디오 서비스의 지연도 많이 줄어든다. 따라서 비디오 시청 수요도 많이 늘어났다. 이로 인해 업로드되는 비디오가 많아짐에 따라 비디오 저장을 위한 스토리지 요구량도 같이 늘어났다.

비디오 스트리밍 업체는 데이터 손실을 막기 위해 스토리지 시스템에서 중복(redundancy) 기법을 많이 사용한다. 중복기술은 크게 복제(replication)과 에러복구코드(erasure code), 2가지 기법이 있다. 복제기법은 단순히 원본 파일을 복제하고 원본 파일의 몇 개 복사본을 더 저장하는 방식이다. 그러나 이 기법이 추가 스토리지 공간이 많이 소요되는 문제점이 있다. 에러복구코드 기법은 데이터를 몇 블록을 나눠서 수학적인 계산을 통해 추가 블록을 만들고 원본 데이터 블록이 손실되어도 남은 블록들을 이용해 데이터를 복구할 수 있는 기법이다. 그러나 에러복구코드는 데이터를 복구하기 위해 여러 개 블록을 읽어야 해서 읽기 대역폭 사용량 많이 필요한 문제가 있다.

스토리지 요구량이 급증하면서 복제 기법의 스토리지 비용이 매우 커지게 되어서 에러복구코드 기법도 많이 쓰게 되었다[1,3,5]. 에러복구코드가 읽기 대역폭이 많은 문제점을 극복하기 위해 많은 연구는 새로운 에러복구코드를 제안했다[1,2]. 새로운 에러복구코드 제안은 기존의 에러복구코드에 비해 성능 개선을 할 수 있다. 그러나 비디오들은 콘텐츠에 따라 인기도와 저장공간이 매우 다양해서 모든 비디오에 한 중복기법을 적용하는 것은 비효율적이다. 따라서 비디오들의 특성을 고려해서 파일별 복제기법을 선택하는지 에러복구코드를 선택하는지 에러복구코드도 어떤 형식인 것을 선택하는지 결정해야 한다.

본 논문에서는 각 비디오의 인기도를 기반으로 비디오 파일의 읽기 대역폭과 추가 데이터 저장 공간의 균형을 고려해서 각각 파일에 대해 최적의 중복 옵션을 선택할 수 있는 휴리스틱 알고리즘을 제안하였다. 우리는 기존에 옵션에서 다른 옵션으로 변경하는 이득을 계산하고, 계산한 값을 사용하여 중복 기법 및 옵션을 결정하는 알고리즘을 개발했다. 실험을 통해 제안한 알고리즘이 비교 알고리즘에 비해 3.32%-21.61% 더 적은 대역폭을 사용하는 것을 확인하였다.

2장에서는 기존의 대역폭이나 스토리지를 고려한 최적화 중복기법에 대한 관련 연구와 중복 옵션을 결정하는 연구를 설명하고, 3장에서는 각각의 파일 별로 중복 기법 및 옵션을 결정하는 문제를 정의하며, 4장에서 우리가 제안한 복제 및 RS코드 옵션 선택 알고리즘을 설명한다. 5장에서는 우리 알고리즘의 성능을 평가하기 위한 실험의 시뮬레이션 환경을 정의하고 실험을 통해 각 파일에 대해 최적의 중복 옵션을 선택하고 정의한 문제를 해결하는 것을 확인한다. 그리고 우리의 알고리즘과 또 다른 2개 기법을 비교하여 우리의 알고리즘의 우수성을 보여준다. 6장에서는 결론을 제시하고 본 논문을 마친다.

2. 관련 연구

스토리지 공간의 최적화를 위한 선행 연구가 수행되었다.

Wang과 그 외 연구진은 접속 요청이 많을 때 복제기법을 쓰고 접속 요청이 적을 때 에러복구코드

를 사용하며 스토리지 공간과 지연을 최적화할 수 있는 기법을 제안했다[12]. Xiang와 그 외 연구진은 에러복구코드, 데이터 배치 및 지연 간 균형을 유지할 수 있는 데이터 배치 기법을 제안했다[13]. 그러나 상기 연구들이 에러복구코드가 소모하는 스토리지 공간과 지연만 고려하고 읽기 대역폭을 고려하지 않았다. Leesakul와 그 외 연구진은 데이터의 중요성에 따라 서로 다른 중복레벨을 정하기와 중복 데이터를 제거하기를 통해 스토리지 공간을 최적화하는 기법을 제안했다[14]. 그러나 연구진은 대역폭에 대해 고려하지 않아 데이터 손실을 방지하는 능력이 부족했다.

대역폭의 최적화를 위한 다양한 기법들이 연구되었다. Duursma와 그 외 연구진은 Reed-Solomon(RS)의 데이터 복구 과정의 개선을 통해 대역폭을 줄이는 기법을 제안했다[11]. Li 와 그 외 연구진은 RS 코딩의 대역폭을 줄일 수 있는 복구 과정을 개선하는 기법을 제안했다[8].

대역폭을 최적화하면서 중복 기법을 수행하기 위한 연구 역시 진행되었다. Nachiappan와 그 외 연구진은, 핫(hot) 데이터에는 복제 방식을 사용하고, 콜드(cold) 데이터는 에러복구코드를 사용하는 혼합 중복 기법을 제안했다[4]. 그들은 대역폭의 최적화를 위해, 콜드 데이터의 경우 고장 발생을 인식했더라고 하더라도 복구를 바로 하지 않고, 나중에 대역폭이 원활할 때 복구하는 lazy 기법을 적용했다.

그러나 상기 연구 역시 데이터의 사용 읽기 대역폭만 고려했을 뿐, 모든 핫 데이터는 복제를 사용하기 때문에, 저장 공간 절약에 대한 고려가 미흡하다.

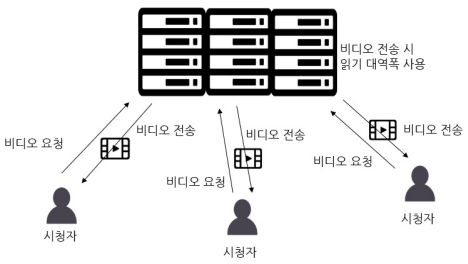
기존의 연구자들은 사용할 중복 기법 및 옵션을 결정하는 이슈에 대한 연구를 수행했다. Kadekodi와 그 외 연구진은, 이질적인 스토리지 시스템에서, 각각의 스토리지의 특성을 고려하여, 각 스토리지에서 사용하는 에러복구코드 옵션을 결정함으로써, 신뢰성을 높이면서 데이터 저장 공간을 절약하는 기법을 제안했다[9]. 그러나 저자들은 스토리지 단위로 중복 기법을 결정했을 뿐, 대역폭에 대해서는 고려하지 않았다. Xia와 그 외 연구진은 서로 다른 2가지 에러복구코드를 사용하며 워크로드에 따라 핫데이터는 복구 성능이 좋은 에러복구코드를 선택하고 콜드 데이터는 필요한 스토리지 공간이 적은 에러복구코드를 사용한다[6].

Li와 그 외 연구진은 데이터의 특성을 맞춰서 핫 데이터가 읽기 대역폭이 적게 필요한 에러복구코드 옵션을 선택하고 콜드 데이터가 읽기 대역폭이 스토리지 공작이 적게 소모하는 에러복구코드를 선택하는 기법을 제안하였다[7]. 그러나 상기 연구들은 에러복구코드만 고려해서 대역폭을 줄일 수 있는 한계가 있다.

우리가 아는 한, 대역폭과 저장 공간을 동시에 고려하면서, 스토리지 단위가 아닌 각각의 파일 별로 중복기법 및 옵션을 결정하는 연구는 기존에 없었다.

3. 문제 정의

서버에 저장된 원본 비디오의 개수를 N^{video}라고 하자. N^{req}는 서버에 도착하는 시청자의 요청 수이며, 서버는 시청자에게 요청된 비디오 $i\,(i=1, \ldots, N^{video})$를 해당 파일이 저장된 스토리지에서 시청자에게 전달한다. 이 때 스토리지에서 비디오 파일들을 사용할 때 읽기 대역폭이 사용된다. (그림 1)은 시청자 요청에 따른 비디오 스트리밍 시나리오이다.



(그림 1) 시청자 요청에 따른 비디오 스트리밍 시나리오

서버는 디스크 고장 시 데이터 손실을 막기 위해 크게 복제과 에러복구코드, 2가지의 중복 기법을

사용한다. 복제는 최소의 대역폭을 사용하지만 지나치게 많은 저장 공간을 사용하므로 적절하게 중복기법을 수행하기 위해서는 에러복구코드 기법의 사용이 필요하다.

본 논문에서 사용한 에러복구코드는 Reed-Solomon(RS) 코딩이다. RS 코딩은 데이터를 여러 블록으로 나눈 후, 데이터 블록을 이용한 계산을 통해 데이터 블록 크기와 같은 크기의 패리티(parity) 블록을 생성하는 기법이다.

모든 비디오는 사용할 중복 기법 및 옵션을 선택해야 한다. N^{option}를 서버에서 고려하는 RS 코딩의 옵션 개수라고 하자. 옵션 j (j=1, ... , N^{option})는 총 블록 개수 n_j와 나누어진 데이터 블록의 개수 k_j의 페어 RS(n_j,k_j)를 의미한다. 이 때 parity 블록이 반드시 존재하므로 $n_j - k_j > 0$이고, 여기서 $n_j - k_j$는 패리티 블록의 개수이다. 예를 들어 서버에서 RS 기법을 사용할 때, RS(4,1), RS(6,3), RS(7,4), RS(8,5)를 사용한다면 N^{option} = 4이다.

R_i은 각 비디오 i가 사용하는 복제 및 RS 코딩의 옵션이다. 옵션 0은 r개의 복제본을 생성하는 복제 기법을 의미한다. 즉 R_i=0이면 복제기법을 사용하고, $R_i \leq 1$이면 RS 코딩를 사용하며, 각각의 옵션은 데이터 블록 및 패리티 블록의 개수가 각각 다르다(i=1, ... ,N^{video} and $R_i \in$ {0, ... , N^{option}}).

중복 기법의 특성상 원본 데이터를 제외하고 추가적으로 생성되는 데이터를 위한 추가 데이터 저장 공간이 필요한데, 이는 선택한 옵션의 값에 따라 달라진다. S^{video_i}를 비디오 i의 크기라고 하자. $S_{i,j}$를 비디오 i가 옵션 j를 선택했을 때 필요한 추가 데이터 저장 공간이라고 하자. S^{limit}를 추가 데이터 저장 공간에 허용되는 저장 공간 한도라고 하자. 모든 비디오에 필요한 추가 데이터 저장 공간의 합 $\sum_{i=1}^{N^{video}} S_{i,R_i}$는 S^{limit}을 초과해선 안된다. $S_{i,j}$는 (식 1)로 계산할 수 있다.

$$S_{i,j} = \begin{cases} S^{video_i} \times r & \text{if } j = 0 \\ S^{video_i} \times \dfrac{n_j - k_j}{k_j} & otherwise \end{cases} \quad (1)$$

비디오의 인기도가 높을수록 비디오 요청의 수가 증가하며, 비디오 요청의 증가는 필요한 대역폭의 증가로 이어진다. p_i를 비디오 i의 인기도라고 하자. 비디오 i에 대한 요청의 개수, N_i^{req}는 $p_i \times N^{req}$로 계산될 수 있다. 우리는 비디오 i가 옵션 j를 선택할 때 사용되는 읽기 대역폭, $B_{i,j}$를 (식 2)와 같이 모델링 한다. 여기서 b^{video_i}는 비디오 i의 파일을 읽을 때에 요구되는 초당 읽기 대역폭이다.

$$B_{i,j} = \begin{cases} b_i^{video} \times N_i^{req} & \text{if } j = 0 \\ b_i^{video} \times N_i^{req} \times k_j & otherwise \end{cases} \quad (2)$$

복제 및 RS 코딩의 각 옵션들은 각각의 장단점이 있으므로, 한 가지 옵션만 사용했을 때 얻는 대역폭 및 저장 공간 측면에서의 이득이 낮다. 따라서 우리는 추가 데이터 저장 공간을 고려하면서, 읽기 대역폭 소모를 최소화할 수 있는 옵션 R_i을 선택하는 문제를 정의하였다. 복제 및 RS 코딩 옵션 선택 문제는 아래와 같이 공식화되었다.

$$\text{minimize} \quad \sum_{i=1}^{N^{video}} B_{i,R_i}$$

$$\text{subject to} \quad \sum_{i=1}^{N^{video}} S_{i,R_i} \leq S^{limit}$$

$$(i = 1, 2, 3 \ldots N^{video})$$

$$(R_i \in \{0, \ldots, N^{option}\})$$

상기 식을 보면 우리의 문제는 다중 선택 배낭문제(multiple choice knapsack) 문제에 해당된다는 점을 알 수 있다. 다중 선택 배낭문제는 무게 제약이 있는 배낭에 몇 그룹으로 나눈 물건들을 그룹 당 최대 1개 물건을 넣을 때 무게 한도를 만족하면서

이득을 최대화하는 방안을 구하는 문제이다. 이 논문의 RS 코딩 옵션 선택 문제는 총 추가 데이터 저장 공간 제약 내에서, 옵션 중 하나를 선택하여, 사용되는 읽기 대역폭을 최소화하는 문제이므로 다중 선택 배낭문제의 일종이다. 다중 선택 배낭문제는 NP-hard이므로 상기 문제도 NP-hard 문제이다.

4. 알고리즘 설계

3장을 통해 복제 및 RS 코딩 옵션 선택 문제가 다중 선택 배낭문제임을 알 수 있다. 다중 선택 배낭문제는 동적 프로그래밍 알고리즘을 이용하여 풀 수 있으나, N^{video}와 N^{option}이 커지게 되면 알고리즘이 필요한 메모리 공간과 계산 시간이 많이 필요하므로 이를 사용하는 것은 비현실적이다. 따라서 본 논문에서는 휴리스틱 기반의 알고리즘을 제안한다.

제안한 휴리스틱 알고리즘은, 기존에 선택된 옵션에서 각 비디오의 옵션을 변경했을 때의 대역폭 및 추가 데이터 저장 공간의 이득을 계산하고 가장 높은 이득을 가지는 옵션을 선택하는 것이 핵심 아이디어이다. 이를 통해 제안한 알고리즘은 크게 2가지 특징을 가진다. 첫 번째는, 전체 비디오의 모든 옵션을 대상으로 고려하여 옵션을 결정하기 때문에, 특정 비디오에 한정된 최적 옵션이 아닌 전체에 대한 최적 옵션을 선택할 수 있다는 것이다. 두 번째는, 비디오의 옵션을 변경할 때마다 해당 비디오가 다른 옵션의 이득을 재계산하기 때문에, 전체 비디오의 상황에 따라 현재 선택한 옵션을 고려하여 최적 옵션을 다시 선택할 수 있다는 점이다. 이러한 특징을 기반으로 제안한 알고리즘을 통해 복제 및 RS 코딩 옵션 선택 문제를 해결할 수 있다.

R_i^{tmp}는 알고리즘에서 지속적으로 변경되는 비디오 i를 위한 임시 옵션 선택 값이다. R_i^{tmp}은 처음에는 복제 기법으로 초기화된다.

휴리스틱 알고리즘에서 이득을 고려하여 옵션을 결정하기 위해 도입한 파라미터 $D_{i,j}$은 다음과 같이 계산된다.

$$D_{i,j} = \frac{B_{i,j} - B_{i,R_{i,j}^{tmp}}}{S_{i,_{i,j}^{tmp}} - S_{i,j}} \quad (3)$$

$B_{i,j} - B_{i,R_i^{tmp}}$는 비디오 i가 기존에 선택한 옵션인 R_i^{tmp}에서 옵션 j로 바뀌었을 때 증가하는 읽기 대역폭이다. $S_{i,R_i^{tmp}} - S_{i,j}$는 비디오 i가 기존에 선택한 옵션인 R_i^{tmp}에서 옵션 j로 바뀌었을 때 감소하는 추가 데이터 저장 공간이다.

읽기 대역폭의 증가를 줄이면서, 추가 데이터 저장 공간을 적게 사용하는 작은 옵션을 선택해야 하므로, 가장 작은 $D_{i,j}$를 선택해야 이득이 높아진다. 이를 위해 우리는 모든 $D_{i,j}$의 값을 오름차순으로 저장한 배열 A를 도입하였다.

(그림 2)는 제안한 알고리즘의 의사코드이다. 본 논문에서 제안한 알고리즘은 다음 설명과 같이 동작한다.

- 초기화 단계 : 각 비디오의 RS 옵션을 가리키는 R_i^{tmp}를 복제 방식의 인덱스인 0으로 초기화한다. S^{tmp}와 B^{tmp}를 모든 비디오들이 옵션 R_i^{tmp}를 선택했을 때 저장할 때 소모하는 읽기 대역폭과 추가 데이터 저장 공간이며, 모든 비디오가 복제 방식을 선택했을 경우의 값으로 초기화한다(1-7번 줄).

- RS 코딩 옵션 선택 단계

(1) 배열 A에서 가장 작은 값인 $D_{m,l}$를 찾고, 배열 A에서 제거한다(8번 줄).

(2) R_m^{tmp}를 가장 큰 이득을 얻는 옵션인 l로 업데이트하고, R_m^{tmp}의 변경에 따라 S^{tmp}와 B^{tmp} 값을 재계산한다(10-12번 줄).

(3) 배열 A에 남아있는, 비디오 m을 위한 모든 옵션 j들의 $D_{m,j}$ 값을 R_m^{tmp}의 업데이트에 따라 업데이트한다(13번 줄).

(4) 배열 A가 비거나, 추가 데이터 저장 공간이 S^{limit}에 도달할 때까지 (1)~(3) 단계를 반복한다(14-15번 줄).

- 마무리 단계 : 각 비디오 i의 옵션 R_i를 R_i^{tmp}로 최종적으로 선택한다(17-19번 줄).

```
Input: B_{i,j} and S_{i,j} (i = 1, ..., N^video and
j = 0, ..., N^option)
Output: R_i (i = 1, ..., N^video)
1 Temporary variables: R^{tmp_i}
2 Array of D_{i,j} values: A
3 for i to N^video do
4       R_i^tmp=0;
5       B^tmp+= B_{i,R_i^tmp};
6       S^tmp+= S_{i,R_i^tmp};
7 end for
8 while(A is not empty) do
9  Find the lowest value D_{i,j} (D_{i,j}∈A) for which
i = m and j = l, and remove D_{m,l} from A;
10      S^tmp = S^tmp − S_{m,R_i^tmp + S_{m,l}};
11      B^tmp = B^tmp − B_{m,R_i^tmp} + B_{m,l};
12      R_m^tmp = l;
13     Update all values of D_{m,j}∈A;
14     if(S^tmp ≦ S^limit)
15          break;
16 end while
17 for i to N^video do
18       R_i = R_i^tmp
19 end for
```

(그림 2) 복제 및 RS 코딩 옵션 선택 알고리즘의 수도코드

5. 실험 결과

5.1. 시뮬레이션 환경

본 논문에서 복제 및 RS 코딩 옵션 선택 알고리즘의 성능을 확인하기 위해, N^{video}=3000일 때에 대한 시뮬레이션을 수행하였다. 각 비디오는 DASH 환경을 지원하기 위해 1초 길이의 청크로 나뉘어져 있으므로, 요구하는 초당 읽기 대역폭은 비디오의 비트율과 같다고 가정한다. 모든 비디오는 8000kbps (즉, 1MB/s)의 비트율을 가지며, CRF로 인코딩되었다고 가정하므로, 비디오는 비디오의 길이에 따라 용량이 달라진다. 예를 들어 3000초 길이의 비디오는 3000MB의 용량을 가지게 된다.

클라이언트의 총 요청수, $\sum_{i=1}^{N^{video}} N^{req_i}$는 10000로 고정되며 모든 비디오의 접근률은 인기도를 따른다. 이는 Zipf 분포로 생성된다. 특별한 언급이 없는 한, Zipf 분포의 디폴트 파라미터는 θ=0.3으로 설정하였다[10]. <표 1>은 실험에서 사용한 옵션 그룹에 대한 상세 내용이다.

〈표 1〉 실험에서 사용된 옵션 그룹

그룹	포함된 옵션 리스트
그룹 1	Replication r=3, RS(7, 5), RS(8, 6), RS(9, 7), RS(10, 8), RS(11, 9)
그룹 2	Replication r=4, RS(7, 4), RS(8, 5), RS(9, 6), RS(10, 7), RS(11, 8)
그룹 3	Replication r=5, RS(7, 3), RS(8, 4), RS(9, 5), RS(10, 6), RS(11, 7)

해당 문제를 해결하기 위한 기존의 기법은 없으므로, 비교를 위해서 추가로 제안하는 알고리즘을 변형하여 StaticD 알고리즘과 Order 알고리즘을 다음과 같이 생성하였다.

(1) StaticD 알고리즘 : StaticD 알고리즘은 우리가 제안한 알고리즘과 흡사한 알고리즘으로, 이득 $D_{i,j}$를 처음에 초기화한 이후 다시는 $D_{i,j}$의 값을 변경하지 않는 기법이며, (그림 2)에서의 13번 줄이 없다. StaticD 알고리즘은 각 비디오의 옵션을 선택했을 때 얻는 이득인 $D_{i,j}$이 제일 높은 것을 선택하는 것은 우리의 알고리즘과 동일하지만, 복제옵션을 기반으로 처음에 계산한 이득 $D_{i,j}$을 알고리즘이 끝날 때까지 이득 $D_{i,j}$을 재계산하지 않음으로써 알고리즘의 계산량을 줄인다. 반면 우리의 알고리즘은 복제 옵션에서 옵션 j로 변경한 후, 이득 $D_{i,j}$를 변경된 옵션을 기반으로 재계산한다.

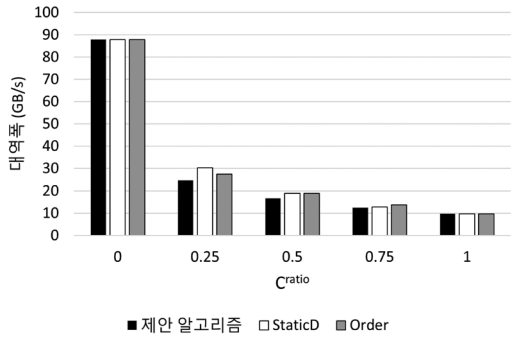
(2) Order 알고리즘 : 제한된 추가 데이터 공간에서 사용하는 대역폭을 최소화하기 위해서 가장 직관적인 방법은 각 비디오가 복제 기법보다 대역폭이 적게 증가하고 공간도 크게 줄일 수 있는 옵션을 선택하는 것이다. Order 알고리즘은 이를 기반으로 설계된 그리디 기반의 기법이다. 먼저 비디오별로 모든 옵션에 대해 $D_{i,j}$값을 계산한다. 다음에 추가 저장 공간 한도를 채울 때까지 인기도가 높은 비디오부터 인기도가 낮은 비디오까지 비디오별로 가장 큰 이득을 가지는 옵션을 선택하기 때문에, 추가 저장 한도가 꽉 차면 인기도가 적은 비디오의 옵션이 선택되지 않은 채로 종료된다. 이는 전체 비디오의 최적을 고려하여 선택하는 우리의 알고리즘과는 차이가 있다.

우리는 제안한 알고리즘과 2가지 비교 알고리즘의 결과를 추가 데이터 저장 공간, 비디오 접근률 분포, 비디오의 크기 및 적용하는 RS 코딩 옵션의 측면에서 비교하였다.

5.2. 추가 데이터 저장 공간 한도에 따른 효과

우리는 추가 데이터 저장 공간 한도에 따른 결과 변화를 확인하기 위한 시뮬레이션을 실시하였다. 실험에서 각 비디오의 길이는 300-9600초 사이이며, Zipf 분포의 파라미터 θ는 0.3이고, <표 1>의 그룹 1을 옵션으로 사용했다.

우리는 추가 데이터 저장 공간 한도에 따른 효과를 비교하기 위해, 새로운 파라미터 C^{ratio}를 도입하였다. $S^{\max}$는 추가 데이터 저장 공간의 최대치이고, 비디오들이 모두 다 복제 기법으로 설정할 때 필요한 추가 데이터 저장 공간이다. $S^{\min}$는 추가 데이터 저장 공간의 최소치이고, $\frac{n_j - k_j}{k_j}$가 가장 작은 RS 옵션을 선택할 때 필요한 추가 데이터 저장 공간이다. S^{limit}는 $(S^{\max} - S^{\min}) \times C^{ratio} + S^{\min}$로 계산되므로, C^{ratio}의 값을 변경하여 S^{limit}을 조절할 수 있다. 즉 C^{ratio}는 추가 데이터 저장 공간의 최대/최소치를 고려한 비율을 결정하는 파라미터로써 [0, 1] 범위의 값을 가진다. C^{ratio}가 0일 때는 $S^{limit} = S^{\min}$, 1일 때는 $S^{limit} = S^{\max}$이다. 만약 모든 알고리즘이 C^{ratio}가 0일 때는 전부 $\frac{n_j - k_j}{k_j}$가 가장 작은 RS코딩 옵션인 RS(11,9)을, C^{ratio}가 1일 때는 복제를 선택하여 대역폭 사용량이 같다.



(그림 3) C^{ratio}의 값에 따른 읽기 대역폭의 변화

우리는 C^{ratio}의 값이 0.25, 0.5, 0.75로 설정하고 결과를 관찰하였다.

(그림 3)은 C^{ratio}의 값에 따라 얼마나 사용된 읽기 대역폭이 달라지는 가를 보여준다. 제안한 알고리즘은 다른 알고리즘에 비해서 언제나 우수했으며 3.32%−18.12% 더 적은 대역폭을 사용하는 것을 보인다. 또한 C^{ratio}가 증가할 수록, 대역폭 사용이 많지만 용량 사용은 많은 옵션을 선택하므로, 사용 읽기 대역폭이 줄어든다는 것을 알 수 있다.

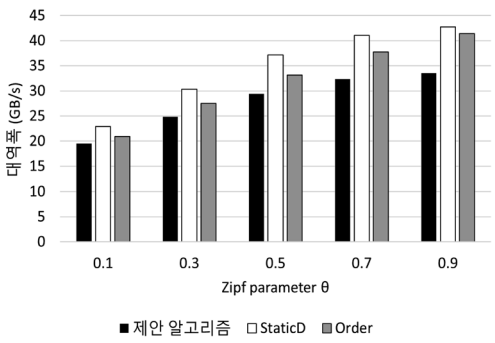
우리는 제안한 알고리즘은 옵션을 선택할 때마다 해당 비디오가 남은 옵션이 현재 선택한 옵션에 대한 이득을 다시 계산하고 전체 비디오 옵션을 고려하기 때문에 타 기법보다 우수한 성능을 보인다. StaticD 알고리즘은 C^{ratio}가 0.25일 때 우리의 알고리즘 대비 가장 낮은 성능을 보이며, C^{ratio}가 0.75일 때 가장 높은 성능을 보인다. 이는 정적 이득 값을 쓰는 StaticD 알고리즘이 옵션 변경 후의 이득을 반영하지 못하기 때문이다. StaticD 알고리즘은 C^{ratio}=0.25일 때는 C^{ratio}가 0.5, 0.75일 때 보다 각 비디오의 옵션을 변경하는 횟수가 많아지면서, 더 많은 오류가 발생하기 때문에 가장 낮은 성능을 보인다. 반면에 Order 알고리즘은 옵션을 선택할 때 비디오가 자기의 옵션의 이득만 고려하기 때문에 전체 최적의 옵션을 선택할 수 없기 때문에 제안한 알고리즘에 비해 좋은 결과를 얻지 못한다.

5.3. 비디오 접근률 분포에 따른 효과

우리는 Zipf 분포의 파라미터, θ에 따른 결과 변화를 확인하기 위한 시뮬레이션을 진행하였다. 실험에서 C^{ratio}=0.25이고, 비디오의 길이는 300−9600초 사이이며, <표 1>의 그룹 1을 옵션으로 사용했다.

θ는 [0,1] 범위의 값으로써 θ가 작을수록 인기도는 더욱 소수의 인기도에 더 집중된다. 만약 θ가 1이라면 모든 비디오의 인기도는 같다.

우리는 zipf 분포의 파라미터 θ가 0.1, 0.3, 0.5, 0.7, 0.9일 때에 대한 실험을 진행하였다.



(그림 4) Zipf 분포의 파라미터 θ에 따른 읽기 대역폭의 변화

(그림 4)는 θ 값에 따라 얼마나 사용된 읽기 대역폭이 달라지는 가를 보여준다. 제안한 알고리즘은 다른 알고리즘에 비해서 언제나 우수하며, 6.74%−21.61% 더 적은 대역폭을 사용한다. 또한 θ의 값이 커질수록 사용한 읽기 대역폭이 증가하는데, 이는 비디오의 접근률이 균등해 질수록, 최대한 용량의 소모를 줄이는 방향으로 읽기 대역폭이 크지만 용량의 소모가 작은 옵션들이 선택되었음을 의미한다. 이는 비디오의 접근율이 균등화될 경우에는 비디오의 인기도의 영향보다는 추가 데이터 저장 공간 한도나 비디오 크기와 같은 다른 환경적 요소의 차이가 결과에 더 큰 영향을 끼친다는 것을 의미한다.

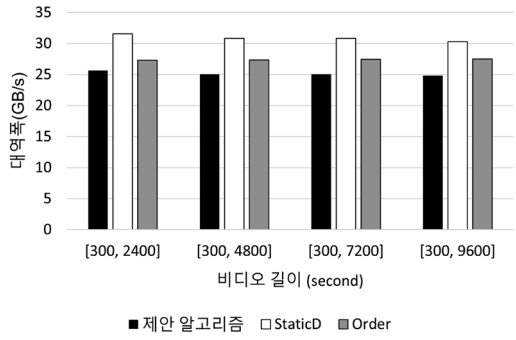
우리의 알고리즘은 타 기법과 비교하여 언제나 θ가 0.9일 때 가장 우수했으며, 이를 통해 비디오의 접근률이 균등할수록 우리의 알고리즘이 우수하다는 것을 확인할 수 있다. 이는 우리가 제안한 알고리즘은 전체 비디오의 모든 옵션의 변화에 대해 잘 고려하기 때문이다. 비디오 인기도의 영향이 약화되고 저장 공간 한도 및 비디오 옵션 크기의 영향이 커져도 우리의 알고리즘은 이를 반영한 선택이 가능하므로, 타 기법보다 언제나 제일 우수하다.

5.4. 비디오 길이에 따른 효과

우리는 비디오의 길이의 차이에 따른 영향을 평가하기 위한 실험을 수행했다. 실험에서 C^{ratio}=0.25이고, Zipf 분포의 파라미터 θ는 0.3이고,

<표 1>의 그룹 1을 옵션으로 사용했다.

우리는 비디오 길이가 300-2400초, 300-4800초, 300-7200초, 300-9600초일 때에 대한 실험을 진행하였다.



(그림 5) 비디오 길이 범위에 따른 읽기 대역폭의 변화

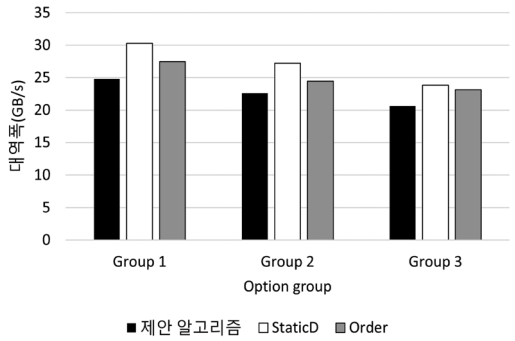
(그림 5)는 비디오 길이의 범위에 따라 얼마나 사용된 읽기 대역폭이 달라지는 가를 보여준다. 제안한 알고리즘은 다른 알고리즘에 비해서 언제나 우수하며, 6.15%-18.81% 더 적은 대역폭을 사용하였다.

우리 알고리즘은 언제나 전체 비디오의 옵션들을 고려하면서 전체 이득이 높아질 수 있도록 각 비디오의 이득을 계산하기 때문에 언제나 제일 우수하다. 우리의 알고리즘과 StaticD 알고리즘은 비디오 길이가 클수록 더 높은 성능을 보이는데, 비디오 길이 범위가 넓어지면 전체 비디오의 옵션 중에서 대역폭의 증가 대비 추가 데이터 공간이 더 많이 줄이는 옵션을 더 많이 찾을 수 있기 때문이다.

반면에 Order 알고리즘은 비디오의 길이의 최대치가 클수록 더 낮은 성능을 보인다. 그 이유는 비디오 길이 범위가 커지면 비디오 크기 역시 증가하면서 저장 공간 제약이 더 증가하는데, Order 알고리즘은 인기도 순서를 따르고 순차적으로 옵션을 변경함으로써 더 많은 비디오의 옵션을 변경하기 때문에, 상대적으로 인기 없는 비디오의 적절한 대역폭 옵션을 선택하기 전에 추가 저장 공간을 꽉 채워 알고리즘이 종료되기 때문이다.

5.5. 옵션 그룹에 따른 효과

우리는 서버에서 사용하는 옵션 그룹이 결과에 주는 영향을 확인하기 위한 시뮬레이션을 수행하였다. 실험에서 C^{ratio}=0.25이고, Zipf 분포의 파라미터 θ는 0.3이고, 비디오의 길이는 300-9600초 사이이다.



(그림 6) 옵션 그룹에 따른 읽기 대역폭의 변화

우리는 <표 1>의 3가지 옵션 그룹을 사용하였으며, 각각의 옵션 그룹에 대한 실험을 진행하였다.

(그림 6)은 옵션 그룹에 따라 얼마나 사용된 읽기 대역폭이 달라지는 가를 보여준다. 제안한 알고리즘은 이번에도 마찬가지로 다른 알고리즘에 비해서 언제나 우수하며, 7.51%-18.12% 더 적은 대역폭을 사용한다. 우리의 알고리즘은 현재의 전체 비디오가 선택한 옵션을 기반으로 각 옵션의 이득을 재계산하여 옵션을 결정할 수 있기 때문에 StaticD 알고리즘보다 우수하다. 또한 Order 알고리즘은 상대적으로 인기 없는 비디오의 적절한 대역폭 옵션을 선택하기 전에 추가 저장 공간을 꽉 채워 종료됨으로써, 전체 비디오의 최적의 옵션을 선택할 수 없기 때문에 우리의 알고리즘은 Order 알고리즘보다 우수하다.

그룹 1이 가장 많은 대역폭을 소모하고, 그룹 3이 가장 작은 대역폭을 소모하는 것을 알 수 있다. 이러한 현상의 이유는 그룹 1의 패리티 블록 숫자

가 2개로 가장 적고, 그룹 3은 4개로 가장 많기 때문이다. 패리티 블록의 숫자가 적을수록 전체 블록 중 데이터 블록의 비율이 증가하기 때문에, 각 옵션들에게 요구되는 대역폭이 증가하면서 전체 대역폭 사용량이 상승하는 것이다.

StaticD 알고리즘은 그룹 3에서 제일 좋은 성능, 그룹 1에서 제일 낮은 성능을 보이는데, 이는 데이터 블록의 비율이 증가할수록 요구되는 저장 공간이 적고 사용되는 읽기 대역폭이 크기 때문이다.

데이터 블록의 비율이 높은 경우, 각 비디오의 옵션을 변경하는 횟수가 많아지면서 더 많은 오류가 발생하기 때문에, 그룹 1에서 가장 낮은 성능을 보인다. 반면 Order 알고리즘은 그룹 1에서 제일 좋은 성능, 그룹 3에서 가장 낮은 성능을 보이는데, 가장 데이터 블록의 비율 큰 옵션 1이, 요구되는 저장 공간이 적고 사용되는 읽기 대역폭을 가장 낮은 인기도에 먼저 할당하는 동작을 수행하는 Order 알고리즘에 적합했던 것으로 보인다.

6. 결론

본 논문에서는 서버에서 비디오 파일 인기도와 추가 저장 공간을 고려하여 파일별 복제 및 RS 코딩 옵션 선택 알고리즘을 제안했다. 네트워크 기술의 발전으로 인해 비디오 스트리밍 서비스가 활성됨에 따라서 비디오 저장 공간 요구량이 크게 증가하였다. 비디오 서비스를 안전하게 제공하기 위해서는 저장된 비디오의 신뢰성을 보장할 수 있는 중복 기법이 필요하다. 중복 기법은 읽기 대역폭이 적고 추가 데이터 공간이 많이 필요한 복제 기법과 읽기 대역폭이 많고 추가 데이터 공간이 적은 에러복구코드로 나눠진다. 이 2개 기법 중에 한 기법만 사용하면 비효율적이므로 파일의 인기도와 추가 데이터 저장 공간, 2개 기법의 장단점을 모두 고려해야 한다.

중복 옵션 선택 알고리즘을 설계하기 위해 우리는 추가 데이터 저장 공간 한도 안에서 각 파일에 적합한 중복 옵션을 선택하는 문제를 정의하였다. 그리고 이 문제를 해결하기 위해 우리는 어떤 파일이 어떤 옵션을 선택할 때 얻을 수 있는 이득 파라미터도 함께 정의하였다.

우리는 설계한 알고리즘의 성능을 검증하기 위해, 실험을 통해 추가 데이터 저장 공간, 비디오 접근률 분포, 비디오의 크기 및 적용하는 RS 코딩 옵션의 측면에서 제안한 알고리즘과 2가지 알고리즘의 결과를 비교하였다. 실험을 통해 우리가 제안한 알고리즘의 우수함을 검증했다.

본 논문의 복제 및 RS 코딩 옵션 선택 알고리즘은 파일별 최적의 중복 옵션을 선택할 수 있는 점을 보여준다.

참고문헌

[1] C. Huang, H. Simichi, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S Yekhanin, "Erasure Coding in Windows Azure Storage.", In Proceedings of the USENIX Annual Technical Conference, Jun. 2012.

[2] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R Vadali, S Chen, "XORing Elephants: Novel Erasure Codes for Big Data", In Proceedings of the VLDB Endowment, Vol. 6, No. 5, pp. 325 - 336, Mar. 2013.

[3] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, V. Sivakumar, L. Tang, and K. Sanjeev. "F4: Facebooks Warm Blob Storage System.", In Proceedings of the USENIX conference on Operating Systems Design and Implementation, pp. 383-398, Oct. 2014.

[4] Rekha Nachiappan, Bahman Javadi, Rodrigo N. Calheiros, Kenan M. Matawie, "Adaptive Bandwidth-Efficient Recovery Techniques in Erasure-Coded Cloud Storage", In Proceedings of the IEEE International Conference on

Parallel and Distributed Computing, pp. 325-338, Aug. 2018.
〔5〕 M. Zhang, S. Han, and P. P. C. Lee,"A Simulation Analysis of Reliability in Erasure-Coded Data Centers", In Proceedings of the IEEE Symposium on Reliable Distributed Systems, Sept. 2017.
〔6〕 M. Xia, M. Saxena, M. Blaum, D. A. Pease,"A Tale of Two Erasure Codes in HDFS", In Proceedings of the USENIX Conference on File and Storage Technologies, Feb. 2015.
〔7〕 J. Li, B. Li,"Zebra: Demand-aware Erasure Coding for Distributed Storage Systems", In Proceedings of the IEEE/ACM International Symposium on Quality of Service, Jun. 2016.
〔8〕 W. Li, Z. Wang, H. Jafarkhani,"A Tradeoff between the Sub-Packetization Size and the Repair Bandwidth for Reed-Solomon Code", In Proceedings of the Annual Allerton Conference on Communication, Control, and Computing, Oct. 2017.
〔9〕 Saurabh Kadekodi, K. V . Rashmi, Gregory R. Ganger, "Cluster storage systems gotta have HeART:improving storage efficiency by exploiting disk-reliability heterogeneity", In Proceedings of the USENIX Conference on File and Storage Technologies, pp. 345-358, Feb. 2019.
〔10〕 Y. Guo, Z. Ge, B. Urgaonkar, P.J. Shenoy, D.F. Towsley, "Dynamic cache reconfiguration strategies for cluster-based streaming proxy", Computer Communications, Vol. 29, No. 10, pp. 1710 - 1721, Jan. 2004.
〔11〕 I. Duursma, H. Dau, "Low bandwidth repair of the RS(10,4) Reed-Solomon Code", In Proceedings of the IEEE Information Theory and Applications Workshop, Feb. 2017.
〔12〕 Y. Wang, J. Wang, M. Wang, J. Liu,"An Adaptive Data Redundancy Strategy in Cloud Storage", In Proceedings of the IEEE International Conference on Electronic Information and Communication Technology, pp. 40-45, Jan. 2019
〔13〕 Y. Xiang, T. Lan, V. Aggarwal, Y. Chen "Joint Latency and Cost Optimization for Erasure-Coded Data Center Storage", IEEE/ACM Transactions on Networking, Vol. 24, No. 4, pp. 2443 - 2457, Aug. 2016.
〔14〕 W. Leesakul, P. Townend, J. Xu, "Dynamic Data Deduplication in Cloud Storage", In Proceedings of the IEEE International Symposium on Service Oriented System Engineering, pp. 320-325, Apr. 2014.

저자소개

◆ 이춘광

- 2020년 인하대학교 컴퓨터공학부 학사
- 2021년~현재 인하대학교 컴퓨터공학부 석사 과정
- 관심분야: 실시간 시스템, 멀티미디어 시스템, 스토리지 시스템

◆ 이다영

- 2016년 인하대학교 컴퓨터공학부(학사)
- 2018년 인하대학교 컴퓨터공학부(석사)
- 2018년~현재 인하대학교 컴퓨터공학부 박사과정
- 관심분야: 모바일 플랫폼, 실시간 시스템, 임베디드 시스템, 멀티미디어 시스템

◆ 송민석

- 1996년 서울대학교 컴퓨터공학과(학사)
- 1998년 서울대학교 컴퓨터공학부(석사)
- 2004년 서울대학교 전기컴퓨터공학부 (박사)
- 2005년~현재 인하대학교 컴퓨터정보 공학부 교수
- 관심분야: 실시간 시스템, 임베디드 시스템, 멀티미디어 시스템