Journal of The Institute of Electronics and Information Engineers Vol.53, NO.1, January 2016 http://dx.doi.org/10.5573/ieie.2016.53.1.097
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2016-53-1-12

파일 분산 저장 시스템의 에너지 효율성 증대를 위한 파일 블록 관리 기술

(File Block Management for Energy-Efficient Distributed Storages)

서 민 국*, 김 성 우**, 서 승 우***

(Min-Kook Suh, Seong-Woo Kim, and Seung-Woo Seo©)

요 약

저장해야 하는 데이터양이 상당히 증가하여 필요 저장 장치의 수가 증가하게 되었다. 다수의 저장 장치 이용 시 일부 저장 장치가 사용 불가능하여도 파일의 가용성을 보장하는 파일 분산 저장 기술이 필수적이다. 최근 다수의 저장 장치로 구성된 파일 분산 저장 시스템의 에너지 소모가 문제가 되고 있다. 파일 분산 저장 시스템의 에너지 효율성을 향상시키기 위한 대표적인 기존 기술은 효율적인 파일 블록 배치를 통하여 사용량이 적은 시간에 일부 저장 장치를 절전 상태로 변경하는 것이다. 한 번 배치가 된 파일 블록의 재배치는 기존 연구에서 고려되지 않는데, 대량의 파일을 저장하고 있는 파일 분산 저장 시스템에서 파일 블록의 재배치에는 큰 비용이 필요할 수 있기 때문이다. 하지만 새 저장 장치나 파일이 추가되는 경우를 고려할 때 파일 블록의 재배치는 필수적이다. 본 논문은 파일 블록의 재배치 시 필요한 블록 이동수를 최소화하는 정수 최적화 문제를 제시하고 이를 분기한정법 알고리즘으로 해결하는 방법을 제안한다. 이를 통해 최소한의 파일 블록 이동으로 최대한의 많은 수의 저장 장치를 절전 상태로 변경할 수 있다. 하지만 정수 최적화 문제의 분기한정법 알고리즘을 통한 해결은 연산 속도가 문제 크기에 따라 지수 함수적으로 증가하는 문제가 있다. 따라서 본 논문에서는 모든 파일과 데이터 서버를 여러 집단으로 나누어 크기가 작은 문제 다수를 해결하는 방식을 제안한다.

Abstract

Because of rapid growth of data size, the number of data storage has been increased. When using multiple data storages, a distribute file system is essential to insure the availability of data files. The power consumption is a major problem when using a distributed file system with many data storages. Previous works have aimed at reducing the energy consumption with efficient file block layout by changing some data servers into stand-by mode. The file block migration has not been seriously considered because migration causes large cost. But when we consider addition of a new data server or file, file block migration is needed. This paper formulates the minimization of data block migration as an ILP optimization problem and solves it using branch-and-bound method. Using this technique, we can maximize the number of stand-by data servers with the minimum number of file block movement. However, computation time of branch-and-bound method of an ILP optimization problem increases exponentially as the problem size grows. Therefore this paper also proposes a data block and data server grouping method to solve many small ILP problems.

Keywords : a distributed storage, energy efficiency, erasure code, ILP, power proportionality

* 학생회원, 서울대학교 전기·정보공학부
(Department of Electrical and Computer Engineering, Seoul National University)
** 정회원, 서울대학교 공학연구원
(Engineering Research Institute, Seoul National University)
** 정회원, 서울대학교 전기·정보공학부
(Department of Electrical and Computer Engineering, Seoul National University)
© Corresponding Author(E-mail: sseo@snu.ac.kr)
※ This paper was supported in part by Samsung Electronics Co., Ltd. This work was also supported by the National Research Foundation of Korea(NRF) grant funded by the Ministry of Science, ICT & Future Planning (MSIP) (No. 2009-0083495).
Received ; October 22, 2015 Revised ; December 14, 2015 Accepted ; December 30, 2015

Ⅰ. 서 론

다수의 파일을 사용하는 어플리케이션과 대용량 파일에 대한 수요가 많아지면서 파일 저장 장치가 저장하는 데이터양이 상당히 증가하고 있다. 이러한 많은 양의 데이터를 저장하기 위하여 다수의 저장 장치를 사용하는 것은 이미 보편화 되었다[1]. 다수의 저장 장치를 사용하는 상황에서는 일부 저장 장치가 고장 나는 상황이 빈번히 발생하기 때문에 파일 분산 저장이 사용된다. 파일 분산 저장 장치란 다수의 저장 장치를 이용, 데이터를 블록으로 나누어 분산 저장하여 데이터의 가용성을 향상시키고 업/다운로드 처리량을 증가시키는 것을 의미한다.

저장하는 데이터양이 저장 장치 용량보다 빠르게 증가하면서 파일 분산 저장 장치를 구성하는 저장 장치의 개수도 증가하였고 파일 분산 저장 장치의 소모 전력이 증가하였다[2]. 또한 하드 디스크와 같은 저장 장치는 요청이 거의 없어 사용 대기 상태로 있는 경우에도 100% 가동되고 있는 경우와 비슷한 양의 전력을 소모하는 특징이 있다[3,4]. 저장하는 데이터양은 증가하였지만 모든 데이터가 높은 사용빈도를 보이는 것은 아니기 때문에 상당한 비율의 저장 장치가 낮은 사용률을 보이게 된다[5]. 이 때문에 파일 분산 저장 장치의 에너지 비효율성 문제가 발생하고 있다.

파일 저장 장치의 하나의 효율성을 증대시키기 위한 연구는 많이 진행되어 왔다[6,7]. 논문[6]에서는 다양한 입력 전압을 가할 수 있는 저장 장치를 이용하여 저장 장치가 대기 상태에 있는 경우 적은 에너지를 사용하게 하였다. 논문[7]에서는 회전 속도를 선택할 수 있는 저장 장치를 이용하였다. 하지만 논문[6]와 논문[7] 모두 현재 사용 중인 저장 장치가 아닌 특수한 저장 장치로 변경해야만 적용 가능하여 많이 사용되지 않았다.

다수의 저장 장치를 사용하는 파일 분산 저장 장치 환경에서는 사용률에 따라 일부 저장 장치를 절전 상태로 변경하여 활성화 상태인 저장 장치의 개수를 변경하는 방식이 연구되었다[8~10]. 이처럼 일부 저장 장치를 절전 상태로 변경하는 방식을 사용하는 경우 모든 파일의 가용성을 보장하는 것이 중요하다.

연산 작업을 분산 처리하는 환경에서는 일부 장치를 절전 상태로 변경하기 위해 연산 작업을 다른 장치로 이동시키는 방식을 이용할 수 있었다[11]. 하지만 수많은 파일을 저장하는 저장 장치에서 파일을 다른 저장 장치로 이동시키는 방식은 비용이 많이 필요하여 사용되지 않았다.

파일 분산 저장 장치에서 파일을 다른 저장 장치로 이동시키지 않고 일부 저장 장치를 절전 상태로 변경하는 방식은 파일 블록을 효율적으로 배치하는 방식이 있다[8~10]. 이 중 논문[8]은 파일 블록 배치를 통해 모든 파일의 수요가 낮아지는 밤에 일부 저장 장치를 절전 상태로 변경할 수 있는 방식을 제안했다.

위와 같이 기존 연구에서 파일 블록 배치를 제안하는 이유는 파일을 이동시키지 않으며 일부 저장 장치를 절전 상태로 변경하기 위함이다. 하지만 모든 파일의 수요는 변화하고 새 저장 장치나 새 파일은 계속 추가된다. 따라서 어떤 고정된 배치도 항상 최적일 수 없고 효율적인 파일 블록 배치의 변경은 에너지 효율성을 증가시킬 수 있다.

이 논문에서는 파일 블록의 배치를 효율적으로 관리하는 파일 블록 관리 기술을 제안하여 파일 분산 저장 장치의 에너지 효율성 증가시킨다. 파일 블록의 재배치 시 필요한 블록 이동을 최소화하기 위하여 정수 최적화 문제로 수식화하였고 최적화 문제를 이용한 파일 재배치 및 서버 관리 알고리즘을 제안한다. 위 기술을 대용량 동영상 스트리밍 서비스나 다수의 파일을 필요로 하는 서비스에 적용하여 시간에 따라 변화하는 사용량에 맞춰 에너지 효율적인 파일 분산 저장 시스템을 구성할 수 있다.

이 논문의 구성은 다음과 같다. 먼저 2장에서 본 논문에서 사용하는 시스템 모델을, 3장에서 파일 블록 관리 알고리즘을 소개한다. 4장에서 시뮬레이션을 통해 알고리즘을 검증하고 5장에서 다수의 저장 장치 및 파일 상황에서의 연산 속도 개선을 위한 기술을 소개한다. 6장에서 결론으로 마무리한다.

Ⅱ. 시스템 모델

1. 시스템 모델

본 논문에서 고려하고 있는 파일 분산 저장 시스템은 그림 1과 같이 데이터를 저장하고 있는 다수의 데이터 서버(데이터 노드, data node)와 데이터 서버를 관리하는 서버 관리자(네임 노드, name node)로 구성되어 있다. 사용자로부터의 파일 요청은 서버 관리자에게 전달되고 서버 관리자가 해당 파일이 저장된 데이터 서버의 위치를 사용자에게 알려주어 파일 전송이 가능하다.

파일 분산 저장 장치에서 사용되는 이레이저 코드(erasure code)[12]는 코드 관리자(coding manager)에서

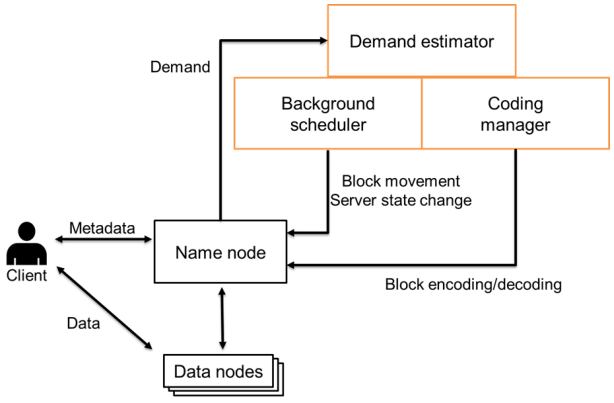

그림 1. 시스템 모델
Fig. 1. system model.

담당한다. 본 논문에서 제안하는 파일 블록 관리 기술(background scheduler)은 서버 관리자를 통해 각 파일 블록이 저장될 데이터 서버를 결정한다.

2. 시스템 가정

본 논문에서는 각 파일에 대한 단위 시간당 요청이 시간에 따라 변하는 상황을 가정한다. 데이터 서버는 단위 시간당 처리할 수 있는 요청에 한계가 있어 각 파일에 대한 단위 시간당 요청이 변화함에 따라 필요한 데이터 서버의 수가 변화하게 된다. 파일마다 단위 시간당 요청량이 다르고, 이는 시간에 따라 변화한다. 각 파일에 대한 단위 시간당 요청은 정확하게 알 수 없으므로 과거 특정 시간 동안의 기록을 통해 측정한다.

각 파일은 (n,k) 이레이저 코드를 통해 n개의 코드화된 파일 블록으로 분할되어 각 데이터 서버에 저장되고 이 중 k개의 코드화된 블록을 이용하면 원본 파일을 복구할 수 있다. 또한 모든 파일의 코드화된 블록의 크기는 같으므로 각 데이터 서버의 저장 용량은 저장할 수 있는 코드화된 블록의 개수로 표현한다.

Ⅲ. 문제 정의

1. 변수 및 정의

본 논문에서 사용하는 문자와 그 정의는 표 1에 정리되어 있다.

2. 파일 및 데이터 서버에 대한 조건

위 변수를 이용하여 각 서버와 파일에 대한 조건을 식 (1)부터 식 (7)로 나타낼 수 있다.

$$\text{minimize} \parallel A - A_0 \parallel \tag{1}$$

표 1. 변수 및 정의
Table 1. Symbols and definitions.

Symbol	Definition
$m \in \mathbb{N}$	전체 데이터 서버의 수
$f \in \mathbb{N}$	전체 파일 개수
$S \in \{0,1\}^{m\times 1}$	서버의 상태 i번째 서버가 활성화 되어 있다면 $s_i = 1$, 절전 상태라면 $s_i = 0$ 이다.
$A \in \{0,1\}^{f\times m}$	파일 저장 위치 i번째 파일의 코드화된 블록이 j번째 서버에 저장되어 있다면 $a_{ij} = 1$ 저장되어 있지 않다면 $a_{ij} = 0$ 이다.
$A_0 \in \{0,1\}^{f\times m}$	현재 파일 저장 위치
$B \in [0,1]^{f\times m}$	해당 파일의 전체 요청 중 해당 서버가 처리하는 요청의 비율
$N \in \mathbb{N}^{f\times 1}$	각 파일의 이레이저 코드 파라미터 n
$K \in \mathbb{N}^{f\times 1}$	각 파일의 이레이저 코드 파라미터 f
$C \in \mathbb{R}_+^{m\times 1}$	각 서버가 저장할 수 있는 코드화된 블록의 개수
$P \in \mathbb{R}_+^{m\times 1}$	각 서버가 단위 시간당 처리할 수 있는 코드화된 블록 요청 수
$D \in \mathbb{R}_+^{f\times 1}$	단위 시간당 각 파일에 대한 요청 수

$$\text{subject to} B \le A \tag{2}$$

$$B \le S \times 1^T \tag{3}$$

$$A \times 1 = N \tag{4}$$

$$A^T \times 1 \le C \tag{5}$$

$$B \times 1 = K \tag{6}$$

$$B^T \times D \le P \tag{7}$$

식 (1)은 본 논문에서 최소화하고자 하는 값으로 전체 파일 블록의 이동수를 의미한다.

식 (2)는 파일이 저장 된 데이터 서버에서만 해당 파일에 대한 요청을 처리할 수 있다는 것을 의미한다. 행렬 A가 0 또는 1 값을 가지므로 $a_{ij} = 0$인 경우 j번째 서버는 i번째 파일을 가지고 있지 않아 해당 파일에 대한 요청을 처리할 수 없다. 따라서 $b_{ij} = 0$이 된다.

식 (3)은 현재 절전 상태가 아닌 활성화된 데이터 서버에서만 파일에 대한 요청을 처리할 수 있다는 것을 의미한다. $s_j = 0$라면 j번째 서버는 현재 절전 상태이다. 따라서 모든 파일 i에 대해 $b_{ij} = 0$이 된다.

식 (4)는 각 파일의 이레이저 코드 파라미터 n을 의

미한다. 각 파일은 코드화된 파일 블록 n개로 분할되었으므로 전체 데이터 서버 중 n개 서버에 해당 파일이 저장 될 것이다. 각 파일 i에 대해 $\sum_j a_{ij} = n_i$이다.

식 (5)는 각 서버에서 저장할 수 있는 저장 용량을 의미한다. 서버 j는 코드화된 블록 c_j개를 저장할 수 있으므로 $\sum_i a_{ij} \le c_j$이다.

식 (6)은 각 파일의 이레이저 코드 파라미터 k를 의미한다. n개의 코드화된 블록 중 k개를 이용해 원본 파일을 복구할 수 있으므로 전체 코드화된 블록에 대한 요청은 원본 파일에 대한 요청의 k배 만큼 들어오게 된다. 따라서 $\sum_j b_{ij} = k_i$이다. 또한 동일한 코드화된 블록 k개를 이용해서는 원본 파일 복구가 불가능하고 서로 다른 코드화된 블록 k개가 필요하므로, 모든 $b_{ij} \le 1$이어야 한다. 이를 해결하기 위해 B 행렬의 모든 원소는 $b_{ij} \in [0,1]$로 범위가 지정되어 있다.

식 (7)은 각 서버가 단위 시간당 처리할 수 있는 요청 개수를 의미한다. 파일 i는 단위 시간당 d_i만큼의 요청을 받고 서버 j는 단위 시간당 p_j만큼의 요청을 처리할 수 있다. 따라서 $\sum_i b_{ij} \times d_i \le p_j$이다.

3. 최소 활성화된 데이터 서버에 대한 조건

모든 데이터 서버가 동일하여 단위 시간당 처리할 수 있는 요청의 수와 저장할 수 있는 코드화된 블록의 개수가 같다고 가정하자. 이 때 해당 시간에 활성화 되어야 하는 데이터 서버의 수는 전체 파일 수와 단위 시간당 모든 파일에 대한 요청을 이용하여 계산할 수 있다.

$$as = \lceil \max\left(\frac{K^T \times 1}{c_0}, \frac{D^T \times K}{p_0}, \max_i(k_i)\right) \rceil \quad (8)$$

식 (8)에서 항 as는 현재 조건에서 얻을 수 있는 활성화된 서버 수의 최솟값을 의미한다. 모든 파일의 가용성을 보장하기 위하여 i번째 파일의 코드화된 블록 중 적어도 k_i개는 활성화된 데이터 서버에 저장되어 있어야 한다. 모든 데이터 서버의 저장 용량이 같다고 가정하면 최소 필요한 활성화된 데이터 서버의 수는 첫 번째 항으로 나타낼 수 있다. 현재 활성화 된 데이터 서버에서 모든 파일에 대한 요청을 처리할 수 있어야 한다. 모든 데이터 서버의 단위 시간당 처리할 수 있는 요청의 수가 같다고 가정하면 최소 필요한 활성화된 데이터 서버의 수는 두 번째 항으로 나타낼 수 있다. i번째 파일을 복구하기 위하여 서로 다른 k_i 개의 코드화된 파일 블록이 필요하다. 따라서 적어도 k_i개의 데이터 서버가 활성화 되어 있어야 하고, 세 번째 항이 이를 의미한다.

Algorithm $filescheduler(D)$
1) $D_0 \leftarrow D$
2) while fm converge
3) $as \leftarrow minstorage(D)$
4) $fm \leftarrow onestepscheduler(D, as)$
5) $D \leftarrow D_0 + |fm| \times 1$
6) $movefile(fm)$

그림 2. 파일 블록 관리 기술
Fig. 2. algorithm filescheduler.

Algorithm $minstorage(D)$
1) return $\lceil \max\left(\frac{K^T \times 1}{c_0}, \frac{D^T \times K}{p_0}, \max_i(k_i)\right) \rceil$

그림 3. 최소 활성화 서버 조건
Fig. 3. algorithm minstorage.

Algorithm $onestepscheduler(D, as)$
1) minimize $\| A - A_0 \|$
subject to $B \le A$
$B \le S \times 1^T$
$A \times 1 = N$
$A^T \times 1 \le C$
$B \times 1 = K$
$B^T \times D \le P$
$S^T \times 1 \le as$
2) return $A - A_0$

그림 4. 파일 및 데이터 서버 조건
Fig. 4. algorithm onestepscheduler.

4. 파일 블록 관리 기술

Ⅲ.2.와 Ⅲ.3. 조건을 모두 만족시키는 파일 블록 관리 알고리즘은 그림 2, 3, 4에 나와 있다.

표 2. 시뮬레이션 환경
Table 2. Simulation parameters.

Parameter	Definition	Value
m	number of data servers	10
f	number of data files	500
K	erasure code parameter n	3
N	erasure code parameter k	5
P	data server performance	15,000 blocks/time
C	data server capacity	300 blocks/server

파일 및 데이터 서버 조건에 Ⅲ.2.절과는 다른 항 하나가 추가되었는데, Ⅲ.3.절에서 계산한 최소 활성화 된 서버에 대한 조건이 추가된 것이다. 이를 통해 파일 블록 관리 기술을 통해 얻은 데이터 서버 상태가 에너지 효율적이라는 것을 보장할 수 있다.

파일 및 데이터 서버 조건을 통해 최적의 서버 상태를 만들기 위해 필요한 파일 블록 이동을 계산할 수 있다. 이 파일 블록 이동은 데이터 서버에게 또 다른 파일 블록 요청이 추가되는 것으로 볼 수 있다. 따라서 동일한 알고리즘을 반복 계산한다.

Ⅳ. 실험 결과 및 고찰

데이터는 논문[8]과 같이 주기적으로 변화하는 파일들의 시간 당 요청을 생성하여 사용하였다. 또한 파일 및 데이터 서버 조건 알고리즘에 필요한 정수 최적화 문제를 계산하기 위한 프로그램으로 MOSEK ApS[13]을 이용하였다. 시뮬레이션에 사용한 파라미터는 표 2에 나와 있다. 모든 데이터 서버는 동일하다고 가정하여 각각 단위 시간 당 처리할 수 있는 요청의 수와 저장할 수 있는 코드화된 파일 블록의 수가 같다. 파일은 이레이저 코드를 통하여 총 5개의 코드화된 블록으로 분할되고 그 중 3개를 이용하여 원본 파일을 복구할 수 있다.

1. 시뮬레이션 결과

이 절에서는 시뮬레이션을 통해 시간에 따라 변화하는 각 파일의 시간 당 요청에 대해 활성화 서버 대수를 유동적으로 조절하여 알고리즘의 효율을 확인하였다. 모든 파일의 요청은 시간에 따라 사인함수 형태로 주기적으로 변화하고, 각 파일의 요청 최댓값은 지수분포를 따른다.

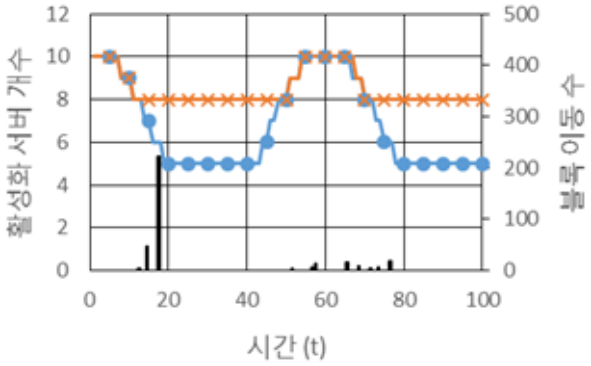

그림 5. 시뮬레이션 결과
Fig. 5. simulation result.

제안한 알고리즘을 통하여 코드화된 파일 블록 배치를 유동적으로 조절하여 활성화 서버 개수를 최소한으로 유지하는 방식은 그림 5에서 파란색 동그라미로 표시 되어 있고 해당하는 파일 이동은 검은색 막대그래프로 나와 있다. 파일 블록 배치의 변경 없이 모든 파일의 가용성을 보장할 수 있는 최소한의 서버 개수는 주황색 엑스 모양으로 표시 되어있다.

2. 결과 분석 및 고찰

코드화된 파일 블록의 배치를 변경시키지 않는 방식의 경우 블록의 초기 배치에 따라 최소한으로 활성화 시킬 수 있는 데이터 서버의 개수가 한정된다. 반면 블록의 배치를 유동적으로 변경하는 방식은 더 적은 수의 활성화 데이터 서버를 얻을 수 있었다.

초기 파일 블록의 배치를 변경하지 않고서는 얻을 수 없던 최소 활성화 서버 대수를 달성하기 위하여 활성화 서버 대수가 변경되는 상황에 블록 이동이 필요한데, 최초로 활성화 서버 대수가 최저가 되는 경우를 제외하면 많은 블록 이동이 필요하지 않았다. 전체 저장 된 블록 개수는 2500개, 그림 5에서 블록 이동이 가장 많았던 경우는 $t=18$에서 222개이다. 이는 전체 저장된 파일 블록 중 8.9%에 해당하는 수치이다.

밤, 낮 하루 주기로 변화하는 파일의 요청을 모델링하기 위하여 각 파일의 요청을 사인함수 형태로 가정하였다. 이와 같은 파일에 대한 요청이 주기적으로 변화하는 상황에서는 파일 블록의 초기 배치를 최적화 하는 데에 필요한 첫 주기에만 많은 블록 이동이 필요하다. 두 번째 주기 이후에는 블록 이동이 거의 없는 것을 확인할 수 있다. 그림 5 $0 \le t \le 30$과 $30 \le t \le 90$에

나타나있다.

시뮬레이션 환경을 변화시켜가며 동일한 실험을 반복한 결과 파일 블록의 배치를 변경하지 않는 방식에 비교하여 파일 블록의 배치를 변경하는 방식은 19.7%의 에너지 효율성 증대 효과가 있었다.

V. 연산 속도 개선 기술

1. 다수의 서버 및 파일 상황 연산량 문제 해결 방안

본 논문에 제안하는 파일 블록 관리 기술에서 가장 연산량이 많은 부분은 파일 및 데이터 서버 조건 알고리즘의 계산에 필요한 정수 최적화 문제이다. 정수 최적화 문제를 분기한정법으로 해결하의 경우 정수형 변수의 개수에 따라 연산 시간이 지수 함수로 증가하게 되는데, 이에 따라 다수의 서버 및 파일 상황에서 연산량 증가 문제를 해결할 수 있는 방법이 필요하게 된다.

파일 및 데이터 서버 조건에서 사용되는 정수형 변수는 서버의 상태를 나타내는 변수 S와 각 파일의 코드화된 블록 저장 위치를 나타내는 변수 A가 있다. 모든 파일은 그 파일의 이레이저 코드 파라미터 n에 맞추어 총 n개의 데이터 서버에 저장 된다. 따라서 이 행렬 A는 행 방향으로 전체 m개 중 n개에만 1 값이 들어가고 나머지는 0으로 채워지는 희소행렬이 된다.

서버의 수 m이 이레이저 코드 파라미터 n과 비교하여 상당히 큰 경우에는 서버 및 파일의 집단화를 통하여 정수 최적화 문제의 연산량을 줄일 수 있다.

$g_1 ... g_v$와 $h_1 ... h_v$에 대하여

(단, $g_1 \cup ... \cup g_v = \{F_1, ..., F_f\}$, $g_1 \cap ... \cap g_v = \varnothing$,

$h_1 \cup ... \cup h_v = \{M_1, ..., M_m\}$, $h_1 \cap ... \cap h_v = \varnothing$)

g_i에 속하는 파일을 h_i에 속하는 데이터 서버에 할당하는 방식이다. 이러한 방식을 이용하면 $f \times m$개의 정수형 변수가 들어가던 하나의 정수 최적화 문제가 $|g_i| \times |h_i|$개의 정수형 변수가 들어가는 v개의 정수 최적화 문제로 바뀐다.

논문[5]에 따르면 집단화를 할 때 전체 데이터 서버의 복구 병렬 처리(rebuild parallelism)를 고려할 필요가 있다. 복구 병렬 처리란 임의의 데이터 서버가 영구적으로 사용 불가능 상태가 되어 새로운 데이터 서버로 대체하는 경우 얼마나 많은 다른 데이터 서버가 병렬적으로 복구에 참여 가능한지를 의미한다. 새로운 데이터 서버를 기존 데이터 서버에 저장되어 있던 코드화된 블록으로 채워 넣는 과정에서 복구 병렬 처리가 좋지 않으면 특정 데이터 서버에만 부하가 집중되어 전체적인 사용률이 낮아지게 된다.

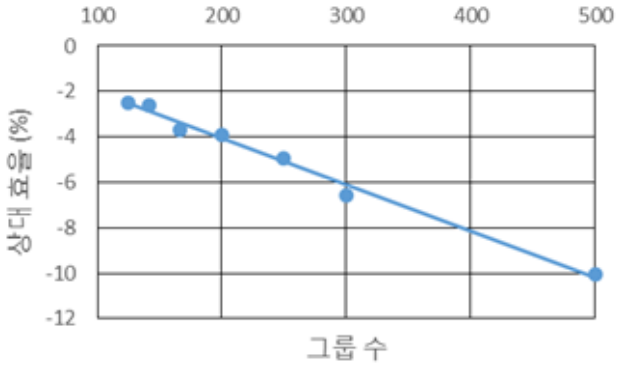

그림 6. 그룹 수와 상대 효율의 관계
Fig. 6. Relation between number of groups and relative efficiency.

서버 및 파일의 집단화를 하는 경우 어떤 $h \in h_i$인 h번째 데이터 서버가 사용 불가능하게 되어 복구가 필요한 경우 h_i에 속하는 데이터 서버만 병렬적으로 복구에 참여 가능하다. 따라서 $|h_i|$는 최소 이레이저 코드 파라미터 n의 수배 이상 되어야 한다.

위와 같이 서버 및 파일의 집단화를 하면 최소 활성화 시킬 수 있는 서버의 수가 다음과 같이 변경된다.

$$\begin{aligned} as &= \lceil \max\left(\frac{K^T \times 1}{c_0}, \frac{D^T \times K}{p_0}, \max_i(k_i)\right) \rceil \\ &= \lceil \max\left(\sum_i k_i/c_0, \sum_i d_i k_i/p_0, \max_i(k_i)\right) \rceil \end{aligned} \quad (9)$$

$$as' = \sum_{i=1}^{v} \lceil \max\left(\frac{\sum_{j \in h_i} k_j}{c_0}, \frac{\sum_{j \in h_i} d_j k_j}{p_0}, \max_{j \in h_i}(k_j)\right) \rceil \quad (10)$$

식 (9)는 집단화 전 최소 활성 서버 수를 의미하고 식 (10)는 집단화 이후 최소 활성 서버 수를 의미한다. 만약 모든 i에 대하여 $\sum_{j \in h_i} k_j$와 $\sum_{j \in h_i} d_j k_j$가 비슷한 값을 갖도록 $g_1 ... g_v$와 $h_1 ... h_v$를 설정한다면 식 (9)과 식 (10)는 큰 오차를 가지지 않게 되어 에너지 효율성의 손해를 보지 않는다.

2. 데이터 서버 및 파일 집단화 시 에너지 효율성의 변화

데이터 서버 3,000대와 파일 1,000,000개를 가정하여 V.1.절에서 제안한 데이터 서버 및 파일의 집단화 기술이 어느 정도의 비용으로 연산량 감소 효과를 얻을 수 있는지 시뮬레이션을 통해 확인하였다.

정수 최적화 문제의 분기한정법을 통한 해법은 정수형 변수의 개수, 즉 한 그룹 당 블록 수에 따라 지수함수로 증가한다. 그룹의 크기를 변화시켜 가며 활성화된 데이터 서버의 수를 확인한 결과는 그림 6에 나와 있다. 세로축은 집단화 전과 비교했을 때의 상대 효율이다.

연산량을 감소시키기 위하여 그룹 수를 늘려 그룹 당 블록 수를 줄이면 필요 활성화 데이터 서버 수가 증가하여 효율 증대 효과가 감소한다. 즉 그룹의 개수는 정수 최적화 문제를 실시간으로 처리할 수 있는 한도 내에서 최소한 작게 정하는 것이 효율적이다.

VI. 결 론

본 논문에서는 파일 분산 저장 시스템의 일부 데이터 서버를 절전 상태로 변경하는 것을 통해 사용량이 적은 경우에도 에너지 효율성을 유지할 수 있는 방식을 제안한다. 데이터 서버 10대와 파일 블록 2,500개를 가정한 시뮬레이션을 통해 확인한 결과 파일 블록의 배치를 변경하지 않던 기존 방식과 비교하여 19.6%의 에너지 효율성 증대 효과가 있었고, 전체 시간 중 가장 많은 블록 배치의 변경이 있는 경우 저장된 파일 블록의 8.9% 정도가 이동되었다.

시뮬레이션을 통해 성능을 검증한 본 연구는 기존 사용되는 파일 분산 저장 시스템에 함께 적용하여 성능을 검증해볼 필요성이 있다. 또한 데이터 서버의 집단화를 통해 연산 속도를 개선한 것 외에 본 논문에서 제시한 정수 최적화 문제를 효율적으로 계산할 수 있는 알고리즘을 제안한다면 더 많은 효율을 얻을 수 있을 것이다.

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S. Leung, “The Google file system,” in SIGOPS Oper. Syst. Rev. vol 37, no 5, pp. 29-43, Oct. 2003

[2] R. T. Kaushik, M. Bhandarkar, and K. Nahrstedt, “Evaluation and analysis of green HDFS: A self-adaptive, energy- conserving variant of the Hadoop distributed file system,” in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, pp. 274-287, Nov. 2010

[3] J. Guerra, W. Belluomini, J. Glider, K. Gupta, and H. Pucha, “Energy proportionality for storage: impact and feasibility,” in SIGOPS Oper. Syst. Rev. vol 44, no 1, pp. 35-39, Mar. 2010

[4] X. Fan, W. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” in SIGARCH Comput. Archit. News, vol 35, no 2, pp. 13-23, June 2007

[5] T. Bostoen, S. Mullender, and Y. Berbers. “Power-reduction techniques for data-center storage systems.” in ACM Comput. Surv. vol 45, issue 3, no 33, pp. 1-38, July 2013

[6] S. Gurumurthiy, A. Sivasubramaniamy, M. Kandemiry, and H. Frankez, “DRPM: Dynamic speed control for power management in server class disks,” in Proceedings of the International Symposium on Computer Architecture (ISCA), pp. 169-179, Jun. 2003

[7] E. Pinheiro and R. Bianchini. “Energy conservation techniques for disk array-based servers,” in Proceedings of the 18th annual international conference on Supercomputing (ICS '04), pp. 68-78, ACM, New York, NY, USA, June 2004.

[8] E. Thereska, A. Donnelly, and D. Narayanan, “Sierra: Practical power-proportionality for data center storage,” in Proceedings of the sixth conference on Computer systems (EuroSys '11), pp. 169-182, ACM, New York, NY, USA, Apr. 2011

[9] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan, “Robust and flexible power-proportional storage,” in Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10), pp. 217-228, ACM, New York, NY, USA, Jun. 2010

[10] B. Lin, S. L, X. Liao, Q. Wu, and S. Yang, “eStor: Energy efficient and resilient data center storage,” in Cloud and Service Computing (CSC), 2011 International Conference on, pp. 366-371, Dec. 2011

[11] H. Goudarzi, M. Ghasemazar, M. Pedram, “SLA-based optimization of power and migration cost in cloud computing,” in Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, pp. 172-179, May 2012

[12] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” in Information Theory, IEEE Transactions on, vol 56, no 9, pp. 4539-4551, Sep. 2010
[13] MOSEK ApS, https://www.mosek.com/

저 자 소 개

서 민 국(학생회원)
2014년 서울대학교 전기정보 공학부 학사 졸업.
2014년～현재 서울대학교 전기정보공학부 석박사통합 박사 과정.
<주관심분야 : 지능형 자동차 지도, 분산 저장>

김 성 우(정회원)
2005년 고려대학교 전자공학과 학사 졸업.
2007년 고려대학교 전자컴퓨터 공학부 석사 졸업.
2011년 서울대학교 전기컴퓨터 공학부 박사
2011～2014년 MIT SMART연구소 박사후연구원
2014년～현재 서울대학교 공학연구원 연구교수
<주관심분야 : 자동차 통신, 대용량 지도관리, 자율주행자동차, 공학교육>

서 승 우(정회원)
1987년 서울대학교 전기공학과 학사 졸업.
1989년 서울대학교 전기공학과 석사 졸업.
1993년 펜실베니아주립대학 전기공학과 박사
1996년～현재 서울대학교 전기컴퓨터 공학부 교수
2009년～현재 서울대학교 지능형자동차IT 연구센터 센터장
<주관심분야 : 자동차IT, 시스템 최적화, 보안>