디스크 배열에서 작은쓰기 문제 해결을 위한 압축 패리티 로깅 기법

김근혜, 장은정, 최황규

강원대학교 컴퓨터공학과

Compressed Parity Logging for Overcoming the Small Write Problem in Redundant Disk Arrays

Geun Hye Kim, Eun Jung Chang, and Hwang Kyu Choi

Dept. of Computer Engineering, Kangwon National University

요 약

본 논문은 RAID5가 갖는 작은 쓰기 문제를 극복하기 위하여 지금까지의 연구 중에서 가장 우수한 성능을 나타내는 패리티 로깅 기법의 단점을 개선한 새로운 패리티 로깅 기법을 제안한다. 제안된 기법은 패리티 로깅기법에서 가장 큰 오버헤드인 패리티 로그의 크기를 줄이기 위하여 로그 데이터를 압축하는 방법을 사용한다. 로그 데이터의 압축은 로그 데이터의 저장에 필요한 비휘발성 버퍼의 크기와 로그 데이터 접근시간을 감소시켜 전체적인 성능을 향상시킬 수 있다. 시뮬레이션을 통한 성능분석 결과에서 제안된 기법은 기존의 패리티 로깅 기법에 비하여 디스크 접근시간에서 우수한 성능을 나타냄을 보인다.

1. 서 론

최근 들어 멀티미디어, 초고속통신망 등의 저변 확대에 따라 컴퓨터 활용분야는 대용량 데이터의 고속, 안정된 운용을 위한 고성능의 보조기억장치를 필요로 하게 되었다. 그러나 대표적인 보조기억장치인 디스크는 기계적인 원인으로 인하여 그 성능향상에 있어 많은 기술적인 어려움을 가지고 있다. 이를 해결하는 방안으로 여러 개의 값싼 디스크를 배열로 구성하고, 에러복구 코드를 추가시키는 새로운 형태의 디스크 시스템 구조인 RAID (Redundant Arrays of Inexpensive Disks)가 출현하였다[1].

RAID는 에러복구 코드와 데이터의 분산방법에 따라 여러 단계로 구분할 수 있으며, 성능 향상을 위한 데이터와 패리티 블록의 위치 결정, 분산단위 크기 연구 등 전반적인 입출력 성능 개선을 위한 노력이 계속되고 있다. RAID 레벨 중에서 가장 일반적인 형태는 RAID5로서 성능과 데이터 신뢰성의 장점으로 인하여 다양한 응용에 활용되고 있다.

그러나 RAID5는 작은 데이터 쓰기(small data write)로 구성된 부하 량에서는 과중한 디스크 억세스로 인하여 처리량이 급격히 감소한다는 단점을 가지고 있다. RAID5에서 한번의 작은 데이터 쓰기를 위해서는 기존의 데이터 읽기, 기존의 패리티 읽기, 새로운 데이터 쓰기, 새로운 패리티 쓰기 등 4번의 디스크 억세스를 요구한다[2] 이를 해결하기 위하여 지금까지 패리티 로깅 기법 등 많은 연구가 진행되고 있다[2-4].

본 논문에서는 RAID5가 갖는 작은 쓰기 문제를 극복하기 위하여 지금까지의 연구 중에서 가장 우수한 성능을 나타내는 패리티 로깅 기법의 단점을 개선한 새로운 패리티 로깅 기법을 제안한다. 제안된 기법은 패리티 로깅기법에서 가장 큰 오버헤드인 패리티 로그의 크기를 줄이기 위하여 로그 데이터를 압축하는 방법을 사용한다. 로그 데이터의 압축은 로그 데이터의 저장에 필요한 비휘발성 버퍼의 크기와 로그 데이터 접근시간을 감소시켜 전체적인 성능을 향상시킬 수 있다.

본 논문은 2장에서 제안하는 압축된 패리티 로깅 기법을 설명하고, 3장에서는 시뮬레이션을 통한 성능분석에 대하여 기술한다. 마지막으로 4장에서는 결론 및 향후 연구에 대해 기술한다.

본 논문은 정보통신부의 정보통신분야 우수학교 지원사업의 지원으로 수행된 연구 결과의 일부임

2. 압축 패리티 로깅 기법

2.1 패리티 로깅 기법

RAID5에서는 실제 대부분의 디스크 사용환경이라 할 수 있는 OLTP(On-Line Transaction Processing)환경에서 지나친 디스크 접근으로 인하여 성능이 저하된다 즉, RAID5에서 한번의 작은 데이터 쓰기를 위해서는 기존의 데이터 읽기, 기존의 패리티 읽기, 새로운 데이터 쓰기, 새로운 패리티 쓰기 등 4번의 디스크 억세스를 요구한다[2].

이러한 문제점을 해결하기 위하여 여러 개의 적은 양의 데이터 쓰기를 모아 큰 데이터 쓰기로 전환함으로써 전체적인 성능을 향상시키려는 노력들이 있었다. 패리티 로깅은 특히 패리티의 수정된 내용을 모아 큰 양의 디스크 억세스로 바꾸고 이를 디스크 배열 상의 로그 영역으로 효율적인 데이터 쓰기를 함으로써 전체적인 패리티를 수정하기 위한 디스크 억세스를 줄여 전체적인 디스크 배열의 성능을 향상시키려는 방법이다[3]

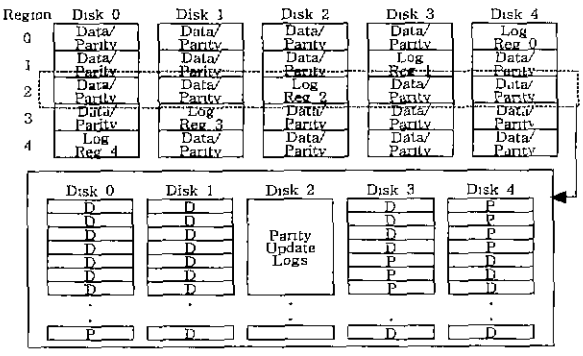

그림 1. 패리티 로깅 디스크 배열 구조

패리티 로깅은 그림 1과 같이 디스크 배열을 데이터 영역, 패리티 영역, 로그 영역으로 나누고 디스크 콘트롤러에 비휘발성 버퍼를 두어 작은쓰기에서 발생한 패리티 수정 정보를 저장한다.

패리티 수정 정보를 위한 비휘발성 버퍼가 다 채워지면 디스크의 로그 영역으로 블록단위의 효율적인 디스크 쓰기를 하며, 패리티 버퍼를 비운다. 그리고 디스크에서 로그 영역이 다 차면 로그 영역내의 패리티 수정 정보와 구 패리티(out-of-data parity) 데이터를 버퍼로 읽어들여 새로운 패리티 데이터를 만든다. 새로 만들어진 패리티 데이터는 패리티 영역에 대용량의 쓰기를 하고 로그 영역은 다음의 패리티 로그를 위하여 비운다. 결과적으로 패리티 로깅에서는 한번의 작은쓰기에 대한 네 번의 디스크 억세스를 두 번의 디스크 읽기와 패리티 로그 정보를 유지하는데 필요한 시간으로 줄일 수 있다.

2.2 압축 패리티 로깅 기법

패리티 로깅 기법은 지금까지 연구된 작은쓰기 문제 해결을 위한 기법들 중 가장 우수한 성능을 나타낸다. 그러나 이 기법은 패리티 로그의 쓰기 및 읽기에 대한 오버헤드를 최소화하기 위하여 패리티 로그 데이터를 가능한 크게 만들어 대용량 쓰기를 해야 하기 때문에 콘트롤러 내에 고가의 대용량 비휘발성 버퍼를 요구한다 또한 한번의 작은쓰기에 대하여 한 블록의 패리티 로그 정보가 발생하기 때문에 패리티 로그 영역이 급격하게 증가하며, 이에 따라 패리티를 재구성하는데 필요한 시간이 증가한다.

본 논문에서는 이러한 문제를 해결하기 위하여 패리티 로그 데이터를 압축하는 기법을 제안한다. 패리티 로그 정보는 구 데이터 블록과 갱신된 새로운 데이터 블록의 Exclusive OR(XOR) 이미지이므로 작은쓰기에서 갱신된 데이터가 블록의 작은 부분에 불과하다는 것을 고려하면 대부분의 데이터는 '0' 값을 가진다. 따라서 적절한 압축 기법을 적용하면 패리티 로그 데이터는 높은 압축률을 얻을 수 있으며, 이에 따라 패리티 로그 데이터 양을 크게 감소시킬 수 있다. 결과적으로 압축 기법을 사용하면 패리티 로깅 기법과 동일한 성능 하에서 비휘발성 버퍼의 양을 크게 줄일 수 있으며, 또한 동일한 버퍼의 양을 사용할 때 패리티 로그 쓰기 및 읽기 횟수를 감소시켜 전체적인 디스크 접근 시간을 감소시킬 수 있다.

압축 패리티 로깅 기법은 그 과정이 기존의 패리티 로깅 기법과 대부분 동일하고, 패리티 로그 데이터를 쓸 때 압축을 수행하고 이를 읽을 때 압축 데이터를 푸는 과정만 추가하면 되므로 매우 간단하다. 본 논문에서는 이 패리티 수정 정보를 압축하기 위해 Run Length Encoding 기법을 사용하였는데, 이는 같은 값이 연속적으로 나타나면 그 값들을 (값, 발생횟수)로 압축하는 방법으로 '0' 값을 많이 갖는 작은쓰기 문제에서 패리티 정보를 압축하기에 적당하다.

3. 성능 분석

3.1 해석적 수식

제안된 압축 패리티 로깅 기법의 성능은 수식적으로 해석할 수 있으며 이에 필요한 파라미터들은 표 1과 같다.

표 1. 시스템 파라미터

S	Average seek time
R	Average rotational delay (1/2 disk rotational time)
H	Head switch time
M	Single track seek time
T	Tracks per cylinder
N	Disks in the array
K	Tracks buffered per region
C	Cylinders per region
D	Data units per track
P	Compression rate ($0 < p <= 1$)

RAID5에서 한번의 작은쓰기를 위해 데이터를 읽고, 이를 갱신하여 디스크에 다시 쓰는데 필요한 시간은 사용자가 원하는 데이터가 존재하는 실린더를 찾는 시간, 헤드 회전 지연시간, 데이터를 읽는 시간, 데이터를 쓰기 위해 다시 한 바퀴를 회전하는 시간, 그리고 데이터를 갱신하는 시간들의 합으로 나타낼 수 있다. 따라서, 이를 위한 평균 디스크 접근 시간은 다음과 같다.

$$\underbrace{(S+R)}_{\text{Seek and Rotational delay}}+\underbrace{2R/D}_{\text{Data prercad}}+\underbrace{(2R-2R/D)}_{\text{Rotational delay}}+\underbrace{2R/D}_{\text{Data write}}$$

이를 간단히 하면

$$S+(3+\frac{2}{D})R=A_0$$

이다 그러나 대부분의 디스크 시스템의 경우 한번 읽은 데이터 블록은 디스크 캐쉬 메모리 상에서 여러 번의 갱신 후에 디스크 쓰기가 이루어지므로 갱신될 데이터 블록의 읽기 시간은 생략될 수 있다. 따라서 한번의 작은쓰기에서 데이터의 갱신에 필요한 디스크 접근 시간은 다음과 같다.

$$S+(1+\frac{2}{D})R=A_0$$

패리티 로그 버퍼에 패리티 수정 정보가 압축률 P로 모두 채워지면 버퍼의 내용은 디스크 쓰기를 통하여 로그 영역으로 저장되고 버퍼는 비워진다. 이때 디스크 접근 시간은 다음과 같다.

$$\underbrace{(S+R)}_{\text{Seek and Rotational delay}}+\underbrace{2RK}_{\text{Data transfer time}}+\underbrace{(K-1)H}_{\text{Head switch time}}$$

여기서 압축률 P를 고려하여 다시 쓰면

$$P(S+(2K+1)R+(K-1)H)=A_1$$

이며, 여기서 압출률은 다음과 같다.

$$\text{압축률 } P=\frac{\text{압축된 패리티 수정 정보 크기}}{\text{원래 패리티 수정 정보 크기}}$$

결국, 로그 디스크 영역이 다 채워지면 패리티 로그들은 패리티들과 다시 재구성됨으로써 로그 영역을 비워준다. 이 패리티 로그들을 재구성하는 것은 로그 영역으로부터 실린더를 읽기, N-1 디스크로부터 패리티 읽기, N-1 디스크로 패리티 쓰기 등의 3단계로 구성된다. 여기서 로그 영역을 읽는데 필요한 디스크 억세스 시간은 다음과 같다.

$$\underbrace{(S+R)}_{\text{Seek and Rotational delay}}+\underbrace{C(2RT+(T-1)H)}_{\text{Read Time for 1 Cylinder}}+\underbrace{M(C-1)}_{\text{C-1 single cylinder seeks}}$$

압축률 P를 고려하여 이를 다시 간단히 쓰면

$$P(S+(2TC+1)R+(T-1)HC+(C-1)M)=A_2$$

이다. 또한 패리티를 읽고 재구성된 패리티를 쓰는데 필요한 시간은 N-1 디스크에서 C/(N-1) 실린더만큼 읽고 쓰는 시간이므로 다음과 같이 나타낼 수 있다

$$\underbrace{(S+R)}_{\text{Seek and Rotational delay}}+\underbrace{(C/(N-1))}_{\text{Cylinders per subaccess}}\underbrace{(2RT+(T-1)H)}_{\text{Read Time for 1 Cylinder}}+\underbrace{(C/(N-1)\cdot 1)M}_{\text{Single track seeks per subaccess}}$$

이를 다시 간단히 쓰면

$$(N-1)(S+R)+C(2RT+(T-1)H)+M(C-N+1)=A_3$$

따라서 작은쓰기를 하는데 걸리는 총 디스크 접근 시간은 다음과 같이 나타낼 수 있다.

$$A_0+\frac{1}{KD}A_1+\frac{1}{DTC}[A_2+2A_3]$$

3.2 시뮬레이션 성능분석

본 논문에서 제안된 압축 패리티 로깅 기법은 시뮬레이션을 통하여 그 성능이 분석되었다. 시뮬레이션을 위한 파라미터는 표 1의 값들을 사용하였다. 시뮬레이션에서 한 데이터 블록에 대한 작은쓰기의 크기는 전체 블록 크기의 100%에서 10%까지의 크기로 가변하여 실험하였으며, 디스크 캐쉬의 효과를 고려하기 위하여 하나의 블록은 1~10번의 작은쓰기 갱신이 캐쉬 상에서 이루어 진 후에 디스크에 쓰기가 이루어지는 것으로 실험하였다.

그림 2는 한 데이터 블록 내에서 데이터 갱신이 1번 이루어졌을 때 작은쓰기의 횟수에 따른 실제 로그 디스크의 쓰기 횟수를 비교한 실험 결과이다. 그림에서 작은쓰기를 수행하는 횟수를 100에서 1000까지로 할 때 압축을 한 경우가 압축을 하지 않고 로그에 쓰기를 하는 기존의 패리티 로깅 기법보다 로그의 쓰기 횟수를 크게 감소시킬 수 있음을 알 수 있다.

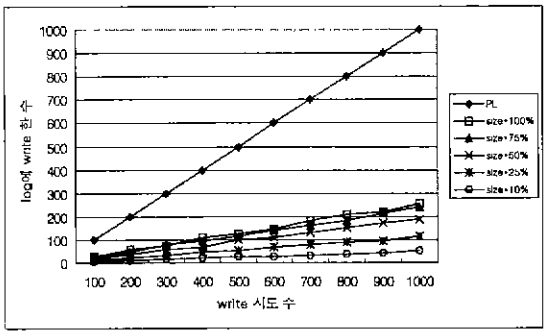

그림 2. 작은쓰기 빈도에 따른 로그 쓰기 횟수

그림 3은 작은쓰기를 시행하는 블록의 수가 1000일 때, 데이터 블록 내의 작은쓰기를 위한 갱신의 크기 변화에 따른 로그의 압축률 변화를 나타낸 것으로, 갱신의 크기가 작을수록 로그 압축률이 감소함을 보인다. 여기서 압축률은 다음과 같다.

$$\text{압축률} = \frac{(\text{원래 로그 크기} - \text{압축된 로그 크기})}{\text{원래 로그 크기}} \times 100$$

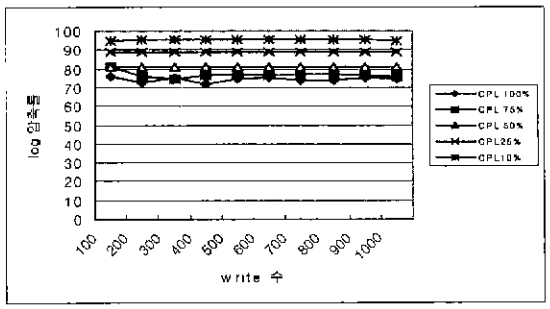

그림 3. 데이터 블록 갱신 크기 변화에 따른 로그 압축률

그림 4는 디스크 캐쉬의 효과를 고려하여 한 데이터 블록에 대해 여러 번의 작은쓰기를 갱신이 일어났을 때 로그의 쓰기 횟수를 비교한 것이다. 1000번의 작은쓰기 시도에서 한 블록에 갱신이 1~10번 일어났을 때 결과에서 갱신의 횟수가 증가하면 압축률이 감소하여 로그 쓰기 횟수가 증가함을 나타낸다.

그림 5는 실험으로 얻은 압축률을 해석적으로 얻은 수식에 적용하여, 작은쓰기 횟수에 따른 디스크 접근시간의 변화를 나타낸 것이다. 압축 기법에 대하여 한 블록에 네의 갱신을 1, 5, 10번 수행한 경우, 갱신의 크기를 블록 크기의 10%로 할 때 압축 기법의 전체 디스크 접근시간은 기존의 패리티 로깅 기법보다 감소함을 보인다.

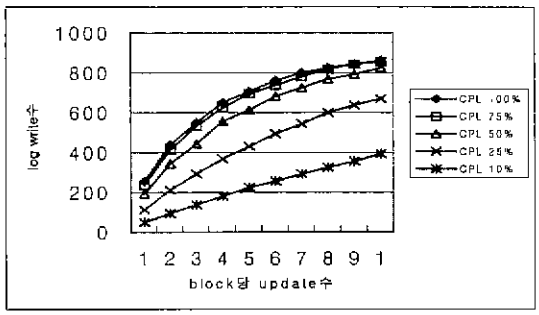

그림 4. 데이터 블록 갱신 횟수에 따른 로그 쓰기 변화

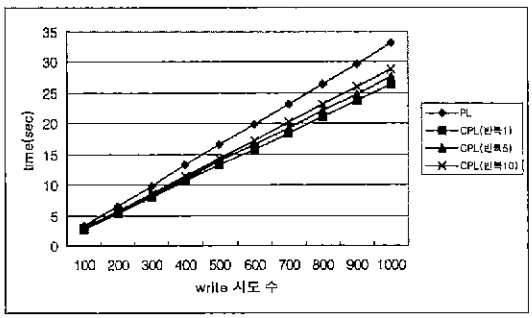

그림 6. Disk busy time

4. 결 론

최근 들어 여러 컴퓨터 활용분야에서 고성능의 보조기억장치의 필요성이 증가함에 따라 RAID가 주목을 받고 있다. RAID의 여러 레벨 중 RAID5는 성능과 데이터 신뢰성의 장점으로 인하여 다양한 응용에 활용되고 있으나 작은 데이터 쓰기로 구성된 부하 량에서는 과중한 디스크 억세스로 인하여 처리량이 급격히 감소한다는 단점을 가지고 있다. 본 논문에서는 RAID5가 갖는 작은 쓰기 문제를 극복하기 위하여 지금까지의 연구 중에서 가장 우수한 성능을 나타내는 패리티 로깅 기법의 단점을 개선한 새로운 패리티 로깅 기법을 제안하였다. 제안된 기법은 패리티 로깅기법에서 가장 큰 오버헤드인 패리티 로그의 크기를 줄이기 위하여 로그 데이터를 압축하는 방법을 사용함으로써 로그 데이터의 저장에 필요한 비휘발성 버퍼의 크기와 로그 데이터 접근시간을 감소시킬 수 있었다. 시뮬레이션을 통한 성능분석 결과에서 패리티 로그 데이터는 매우 높은 압축률을 보였으며, 기존의 패리티 로깅 기법에 비하여 디스크 접근시간에서 우수한 성능을 나타냄을 보였다.

참고문헌

[1] K. Salem and H Garcia-Milina, "Disk Striping," Proc. of 2nd Int. Conf. on Data Engineering, 1986.

[2] P. Chen, E. Lee, G Gibson, R. Karz, and D. Patterson, "RAID: High-Performance, Reliable Storage Architecture." ACM Computing Surveys, June 1994.

[3] D. Stodolsky, G. Gibson, and M. Holland, "Parity Logging Overcoming the Small Write Problem in Redundant Disk Arrays," Proc. of Int. Symp. on Computer Architecture, 1993.

[4] E. Gabber and H. Korth, "Data Logging' A Method for Efficient Data Updates in Constantly Active RAIDs," Proc. of 15th Int. Conf. on Data Engineering, 1998