Journal of the Korea Academia-Industrial cooperation Society
Vol. 16, No. 4 pp. 2784-2790, 2015

http://dx.doi.org/10.5762/KAIS.2015.16.4.2784
ISSN 1975-4701 / eISSN 2288-4688

컬럼-기반 데이터베이스를 위한 그림자 복구

변시우[1*]
[1]안양대학교 디지털미디어학과

Shadow Recovery for Column-based Databases

Sl-Woo Byun[1*]
[1]Division of Digital Media, Anyang University

요 약 컬럼-기반 데이터베이스 저장소는 우수한 입출력 성능으로 대용량 데이터 트랜잭션을 위한 매우 진보적인 모델이다. 전통적인 데이터 저장소는 빠른 쓰기 연산을 위하여 한 레코드의 속성들을 하드 디스크에 연속적으로 배치되어 있는 가로-지향 저장 모델을 활용하였다. 하지만 검색이 대부분인 데이터웨어하우스 시스템을 위해서는 월등한 판독 성능 때문에 컬럼-지향 저장소가 더 적합한 모델이 되고 있다. 또한 최근에는 플래시 메모리를 사용한 SSD가 고속 데이터 분석 시스템을 위한 적합한 저장 매체로 인식되고 있다.
본 연구에서는 플래시 미디어 파일 시스템을 기반으로 하는 컬럼-기반 데이터베이스 환경을 위한 새로운 트랜잭션 회복기법(CoSR)을 제안한다. 제안 기법은 기존의 쉐도우 페이징 기법을 개선하여 플래시 파일 시스템에서 새로운 블록에 데이터를 저장할 경우 무효화되어 폐기되는 이전 데이터 블록을 재활용하였다. 이를 위하여 제안된 컬럼-기반 쉐도우 복구 기법에 재활용 쉐도우 리스트 구조를 활용하였다. 제안 기법은 기존 쉐도우 페이징기법의 최대 단점인 쉐도우 페이지 관련 추가 저장 공간의 부담을 최소화하고, 기존 복구 기법에서 컬럼 데이터 압축에 기인한 입출력 성능저하를 최소화 할 수 있다. 실험 분석 결과를 통하여 CoSR기법이 기존 기법보다 17% 더 우수함을 확인하였다.

Abstract The column-oriented database storage is a very advanced model for large-volume data transactions because of its superior I/O performance. Traditional data storages exploit row-oriented storage where the attributes of a record are placed contiguously in hard disk for fast write operations. However, for search-mostly data warehouse systems, column-oriented storage has become a more proper model because of its superior read performance. Recently, solid state drive using flash memory is largely recognized as the preferred storage media for high-speed data analysis systems. In this research, we propose a new transaction recovery scheme for a column-oriented database environment which is based on a flash media file system. We improved traditional shadow paging schemes by reusing old data pages which are supposed to be invalidated in the course of writing a new data page in the flash file system environment. In order to reuse these data pages, we exploit reused shadow list structure in our column-oriented shadow recovery(CoSR) scheme. CoSR scheme minimizes the additional storage overhead for keeping shadow pages and minimizes the I/O performance degradation caused by column data compression of traditional recovery schemes. Based on the results of the performance evaluation, we conclude that CoSR outperforms the traditional schemes by 17%.

Key Words : column-oriented database system, database recovery, flash memory, reused shadow list

1. 서론

컬럼-지향(Column-Oriented) 데이터베이스[1,2]는 세로의 필드 단위로 분리, 저장, 검색하는 새로운 데이터베이스이다. 판독 위주의 데이터베이스 환경에서는 특정 컬럼에 데이터가 모여 군집되어 있는 컬럼 저장 구조가

*Corresponding Author : Siwoo Byun(Anyang Univ.)
Tel: +82-31-467-0922 email: swbyun@anyang.ac.kr
Received January 18, 2015 Revised March 9, 2015
Accepted April 9, 2015 Published April 30, 2015

당연히 효율적이며, C-Store[3,4]가 잘 알려져 있다. 또한 최근에는 전통적인 하드디스크 저장 시스템에서 SSD 저장 시스템으로의 대체이동이 빠르게 진행되고 있다.[5,6]

기본적으로 세로-저장 스토리지는 테이블 저장시 세로로 분리 저장한다. 예로 들어, 논리적으로 행과 열의 2차원으로 구성되어 있을 경우, 실제 물리적인 저장은 해당 저장 장치에서 연속적인 일차원 바이트 열로 구성되어 운영체제나 버퍼와 연동된다.[7]

물론, 컬럼-지향 데이터베이스의 저장 장치로서 기존의 전통적인 하드디스크를 그대로 사용할 수도 있다. 하지만, 본 연구에서처럼 새로운 저장기술인 플래시 메모리 저장 장치를 최신 컬럼-지향 데이터베이스에 접목시키면, 다음과 같은 이유로 최상의 효율과 성능을 만들 수 있다.

먼저, 플래시 메모리 SSD는 그동안 고가였던 이유로 검토 대상에서 제외되었지만, 충분한 가격경쟁력을 갖추고 있음이 첫째 이유이다. 둘째, 플래시 메모리가 대용량화 되고 있으며, 판독 속도가 하드디스크에 비하여 매우 빠른 장점이 있다. 따라서, 세로-저장 스토리지의 사용처가 대부분 고속 읽기가 필요하므로 매우 적합하다.

이러한 관점에서, 본 연구에서는 컬럼-지향 데이터베이스의 운영 특성을 효과적으로 활용하며, 고속 플래시 메모리 SSD의 장점을 최대한 반영하고자 한다. 또한, 높은 부하와 변동성에서도 대용량 고속의 컬럼-지향 데이터베이스에 최적화된 트랜잭션 처리 성능과 안정성을 확보하기 위하여, 컬럼-데이터의 압축 특성과 플래시 메모리 파일 시스템의 특성을 효과적으로 활용한 새로운 컬럼-지향 데이터베이스 회복 기법을 제안한다.

2. 일반 데이터베이스의 복구 방법론

일반적으로 데이터베이스 시스템에서는 트랜잭션 관리자의 하부에 데이터 관리자가 연동되어 있으며, 데이터 관리자는 내부에 회복 관리자를 거쳐서 실제 스토리지에 접근하므로, 평상시에도 회복 기능이 원활히 작동되어야 한다. 이를 위하여 데이터베이스 관리 시스템은 트랜잭션이 수행되는 동안 데이터 변경에 대한 정보를 지속적으로 유지하여야 한다[8].

트랜잭션의 수행 실패와 같은 비재해적 고장에서 데이터베이스를 회복하는 기법에는 즉시 갱신(update-in-place) 기반[8,9]과 쉐도우 페이징(shadow paging) 기반[10,11]이 있다. 즉시 갱신 기법은 수정된 데이터 항목을 디스크와 같은 저장 장치로 반영(flush)할 때, 같은 위치에 직접 기록하여 이전의 데이터를 덮어쓰므로, 각 데이터 항목에 대하여 하나의 사본이 존재하며, 회복을 위한 로그들을 반드시 유지해야 한다. 반면에, 쉐도우 페이징 기법에서는 새로 저장되는 데이터 항목은 디스크 저장 장치의 다른 위치에 저장하므로 같은 데이터 항목에 여러 개의 사본이 존재 가능하다. 쉐도우 페이징 기법은 이전 값과 이후 값을 모두 디스크에 보관하므로 즉시 갱신 기법과는 달리 회복을 위하여 로그를 유지할 필요가 없는 장점이 있다.

일반적으로 즉시 갱신 기법은 redo/undo 로그 저장에 따른 오버헤드로 시스템의 성능 저하가 공통적으로 발생하는 반면에, 쉐도우 페이징 기법에 비하여 저장 공간이 적게 소모되는 장점이 있다. 쉐도우 페이징 기법은 트랜잭션이 성공적으로 수행 완료될 때까지 다른 영역에 수정분을 보관하고 있다가 완료 시점에 수정분이 데이터베이스에 반영된다. 따라서 오손 페이지(dirty page)가 발생하지 않고, 트랜잭션의 redo/undo 연산이 불필요하므로 관련된 로그의 부담이 없으며, 트랜잭션 실패 후 회복 작업이 매우 간편하다. 반면에 쉐도우 페이지를 보관하기 위한 많은 저장 공간이 필요하며, 이를 효율적으로 관리하기 위한 시스템 오버헤드가 수반된다. 또한, 갱신된 데이터베이스의 페이지가 디스크 공간상에서 위치가 자주 바뀌므로, 연관된 페이지가 집중되지 않고 분산되므로 디스크 입출력 성능이 저하되고, 결과적으로 데이터베이스 시스템의 처리 성능이 저하된다[12].

3. 컬럼-지향 복구 기법의 제안

3.1 플래시 메모리 접근 및 처리 구조

제안 시스템의 처리 구조를 살펴보면 그림1과 같다. 즉, 컬럼-지향 데이터베이스 관리자, 플래시 메모리 페이지 관리자, 플래시 메모리 세그먼트 관리자를 근간으로 구성된다.

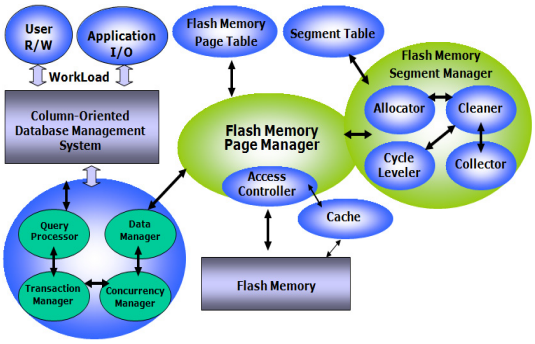

Fig. 1. Flash-based Column-Oriented Database

컬럼-지향 데이터베이스 관리자는 사용자로부터 트랜잭션 처리 요청을 받아서, 트랜잭션 관리자, 쿼리 처리자, 동시성 관리자, 데이터 관리자의 도움으로 트랜잭션의 수행 완료까지의 전체적인 과정을 담당한다. 플래시 메모리 페이지 관리자는 매핑 테이블과 세그먼트 테이블을 관리하며, 접근 제어기를 통하여 요청된 데이터가 플래시 저장소에 안전하게 저장되도록 관리한다.

플래시 메모리 세그먼트 관리자는 부수적으로 allocator, cleaner, cycle leveler, collector의 4개의 모듈을 포함한다. Allocator는 프리 세그먼트들의 풀(Free Segment Pool)을 유지하며, 어떤 세그먼트가 다음에 할당되어야 하는지를 결정한다. Cleaner는 프리 세그먼트(페이지)를 생성하기 위하여 무효화된 페이지를 제거하는 작업을 수행한다. Cycle Leveler는 전체적인 플래시 메모리의 세그먼트들이 균등하게 소거되도록 조절하는 작업을 수행한다. Collector는 콜드 데이터와 핫 데이터를 분리하는 작업을 수행하여 Cleaner의 작업 부하를 경감시켜준다[13].

3.2 플래시 메모리 파일 시스템의 구조

플래시 메모리는 어느 특정 블록에 집중하여 사용하면, 수명이 다하여 나머지 다른 부분도 사용할 수 없게 된다. 따라서 전체 영역에 걸쳐서 비슷한 비율로 데이터 기록이 분산되어야 한다. 이러한 특성 때문에 플래시 메모리를 사용하는 파일 시스템에서는 로그-구조 파일 시스템[14]을 근간으로 발전되었다. 플래시 메모리를 사용하는 임베디드 시스템 분야에서 많이 사용하는 파일 시스템은 JFFS(Journaling Flash File System)에서 개선된 JFFS2이다.[15] JFFS는 일반적인 파일 시스템을 개선하여 플래시 메모리의 특성을 고려한 파일 시스템이다.

본 연구에서는 이러한 특성의 플래시 파일 시스템에서 새로운 블록(페이지)에 데이터를 저장할 경우 무효화되어 폐기되는 이전 데이터 블록을 효과적으로 재활용하게 한다. 즉, 컬럼-지향 데이터베이스의 트랜잭션 회복시 그림자 페이지로 재활용하여, 시스템의 성능을 향상시키고 안정성을 높이고자 한다.

3.3 컬럼-지향 쉐도우 복구(CoSR) 기법

기존의 쉐도우 페이징 기법은 트랜잭션이 실행을 시작할 때, 현재 페이지 테이블을 쉐도우 페이지 테이블로 복사한다. 현재 페이지 테이블의 목록은 가장 최근의 데이터베이스 페이지들을 가리키며 트랜잭션이 수행되면서 수정되는 반면, 쉐도우 페이지 테이블은 트랜잭션이 실행되는 동안에 절대 수정되지 않고 유지된다. 즉, 쓰기 연산 실행시 새로운 페이지 사본이 생성되지만 그 페이지의 이전 사본을 덮어쓰지 않는다. 따라서 트랜잭션이 갱신한 페이지들에 대하여 두 개의 버전이 유지되는데, 이전 버전은 쉐도우 페이지 테이블이 참조하고 새 버전은 현재 페이지 테이블이 참조한다.

만일 데이터베이스 운영중에 트랜잭션 실패가 발생하여 복구해야 한다면 현재 페이지 테이블을 버리고 이전의 쉐도우 페이지 테이블을 현재 페이지 테이블로 교체하면 된다. 반대로 트랜잭션이 성공적으로 수행되어 정상적으로 완료되면 쉐도우 페이지 테이블을 더 이상 필요가 없으므로 폐기하면 된다. 이 과정이 쉐도우 페이징 기법의 기본적인 운영 방식이며 redo/undo 연산이 없으므로 no-redo/no-undo 기법으로 분류된다[8].

본 논문에서 제안하는 플래시 메모리 기반의 컬럼-지향 쉐도우 복구 기법은 플래시 메모리 저장 특성을 효과적으로 활용한 새로운 트랜잭션 회복 기법이다. 즉, 기존의 쉐도우 페이징 기법의 큰 장점인 간편한 처리 구조는 유지하되, 큰 단점인 쉐도우 페이지를 저장하기 위한 별도의 저장 공간 오버헤드를 제거하여 컬럼-지향 데이터베이스 시스템의 성능과 안정성을 향상시키는 것을 목표로 한다.

이를 위하여 앞 절에서 설명한 플래시 파일 시스템의 특성을 효율적으로 활용해야 한다. 즉, 플래시 메모리의 물성적인 특성을 보완하기 위하여 거의 대부분의 플래시 파일 시스템이 사용하는 LFS 기반 방식에서 한 번 저장 후 폐기되는 데이터 블록(페이지)을 재활용해야 한다. 전술한 바와 같이 가장 보편적인 JFFS와 같은 LFS 기반 플

래시 파일시스템에서는 새로운 블록에 데이터를 저장할 경우 이전 데이터 페이지는 무효화된 후 폐기된다. 이러한 무효화된 페이지의 재활용을 위해서는 기존 파일 시스템과의 연동, 재활용 쉐도우 리스트(Reused Shadow List) 구조, 트랜잭션 회복 관리가 필요하다.

여기서 폐기되는 쉐도우 페이지의 활용효과는 플래시 메모리 파일 시스템이 데이터들로 완전히 차지 않고, 어느 정도 여유 공간이 있다는 가정을 전제하고 있다. 예를 들어 파일 시스템 풀이 발생한다면, 쉐도우 페이지로 갈 여유 블록이 없으며, 바로 클리닝 과정을 거쳐서 새로운 블록으로 할당시켜야 하기 때문에 페이지 활용효과는 없어지고, 오히려 오버헤드만 증가한다. 다만, 컬럼-지향 데이터베이스가 대용량 데이터 분석 용도로 쓰기 보다는 읽기 트랜잭션의 비중이 훨씬 더 크며, 낮은 빈도의 쓰기 트랜잭션 중에서, 뒤로 롤백되는 빈도는 더욱 낮다. 또한, 일반적으로 플래시 파일 시스템은 최소 5% 정도는 여유 공간을 유지하므로 오버플로우가 평상시에는 발생하지 않는다. 만일, 오버플로우시에는 재활용 쉐도우 리스트의 해당 페이지를 재활용 쉐도우 리스트에 넣지 않고, 기존 기법처럼 일반 디렉토리하에 위치한 파일처럼 해당 페이지가 저장되게 된다.

제안된 CoSR 기법을 적용하면 갱신 연산을 로깅 오버헤드 없이 처리 가능하다. 또한, 로깅에 대한 지연과 체크포인팅 부담이 제거되므로 결과적으로 트랜잭션의 응답성능과 처리 성능을 높일 수 있다. 앞 절에서 설명한 쉐도우 페이징 기법의 단점을 재검토해 보면 다음과 같다.

첫째, 기존 기법에서는 쉐도우 페이지를 보관하기 위한 많은 저장 공간이 소모된다는 단점이 있다. 하지만, CoSR기법에서 이 단점은 플래시 파일 시스템에서 버려지는 과거 데이터 페이지를 쉐도우 페이지로 재활용하기 때문에 추가적인 공간이 낭비되지 않는다. 따라서 쉐도우 페이지를 위한 공간 부담이 제거되어 이 단점은 극복된다.

둘째, 디스크 기반 파일 시스템 환경에서는 쉐도우 페이징이 진행될수록 갱신된 데이터 페이지가 디스크 저장 공간상에서 페이지 위치가 자주 바뀌는 단점이 있다. 이는 연관된 페이지가 집중되지 않고 넓게 분산되므로 입출력 성능이 저하되고, 결과적으로 데이터베이스 시스템의 처리 성능이 저하된다. 그러나 CoSR환경에서는 디스크가 아닌 플래시 메모리를 저장 장치로 사용하므로 메인 메모리와 같은 방식의 신속한 임의 접근이 가능하다. 따라서 데이터 페이지 분산에 의한 입출력 성능저하가 발생하지 않으므로 이 단점도 극복된다.

셋째, 기존 즉시 갱신 기법과 비교해 보면 즉시 갱신 기법은 롤백시 언두 연산후에 컬럼-데이터의 압축 전략과 판단 과정을 거치고, 압축인코딩한 후 저장하여야 하는 과정에서 많은 시간이 소모된다. 하지만 CoSR 기법은 이미 압축 저장된 이전 이미지 페이지로 링크를 되돌리면 되므로 상대적으로 시간이 절약되며 부수적으로 플래시 메모리에 저장하는 입출력이 감소된다.

결과적으로 제안된 CoSR기법은 즉시 갱신 기법에서 발생하는 롤백시의 압축 부하와 저장 부하를 감소시키며, 플래시 파일 시스템에서 버려지는 데이터 페이지를 재활용함으로써 기존 쉐도우 페이징 기법의 저장 공간 낭비를 최소화하여, 데이터베이스 시스템의 효율과 성능을 높일 수 있다.

4. 부하 생성 및 성능 분석

4.1 실험 시스템 모델

본 실험에서 사용된 기법은 제안기법인 CoSR, 즉시 갱신 기법인 UiPR과 기존 쉐도우 페이징 기법인 BaSR이며, 실험 도구는 Win7 서버에서 CSIM[16]시뮬레이션 언어와 비주얼 C++를 사용하였다.

실험 시스템은 부하 생성기(Workload Generator: WG), 컬럼-지향 데이터베이스의 트랜잭션 관리자(CoTM), 컬럼-지향 데이터 관리자(CoDM), 플래시 페이지 관리자(FPM)로 구성된다.

부하 생성기는 시뮬레이션에 필요한 작업 부하를 만들기 위하여 특정 간격으로 컬럼-지향 데이터베이스에 접근하는 사용자 트랜잭션을 생성시킨다. 트랜잭션 관리자는 사용자 트랜잭션의 생성부터 종료까지의 수행을 관리하며, 트랜잭션을 분석하여 데이터 관리자의 데이터 접근 요청큐에 보낸다. 데이터 관리자는 논리적인 데이터 접근을 수행한 후, 물리적인 접근 요청은 플래시 페이지 관리자에게 보낸다. 플래시 페이지 관리자는 재활용 쉐도우 페이지를 관리하며 입출력 요청을 수행한다.

시뮬레이션의 주요 성능 평가 지표는 저장 공간 소모량(Space Consumption), 트랜잭션 처리치(throughput)와 응답시간(response time)이다.

4.2 실험 데이터 모델

본 실험에 앞서서, 총 1억 건의 사전에 구축한 데이터를 사용하였다.[2] 컬럼 데이터를 압축 저장시 대용량의 컬럼 데이터를 통째로 저장하면, 압축 시간도 매우 느리고, 복원 시간도 느리게 되므로, 적당한 양의 세그먼트로 단위로 나누어서 저장하게 하였다.

4.3 실험 결과 및 분석

실험을 통하여 UiPR, BaSR, CoSR의 성능을 분석하였다. CoSR15는 기본 CoSR 기법에서 재활용 리스트의 크기를 기본 30개에서 15개로 낮추어 재활용이 어려운 여건에서 수행한 결과이다. 실험에서 사용자 트랜잭션의 수를 변화시킴으로써 각 기법의 작업부하의 변화를 살펴보았다.

그림2에서, 전체 구간에서 쉐도우 페이징 기법인 CoSR, BaSR기법의 그래프가 UiPR 그래프 보다 상단에 위치하므로, 트랜잭션 처리 결과가 더 우수함을 알 수 있다. 특히, 컬럼 데이터의 속성이 텍스트로서 주소 속성과 같은 불규칙한 문자열이 많이 포함되는 경우는 27%로 압축되어 상대적으로 큰 부하를 발생시키고 이는 UiPR 기법에 불리하게 작용하게된다.

그림3에서도 트랜잭션의 초당 발생량이 증가할수록, 작업부하가 늘어나므로 응답시간도 서서히 증가함을 알 수있다. 두 기법의 그래프를 비교해 보면 전체 구간에서 평균 응답시간이 CoSR기법이 압축부하가 큰 UiPR 기법보다 더 빠름을 알 수 있다.

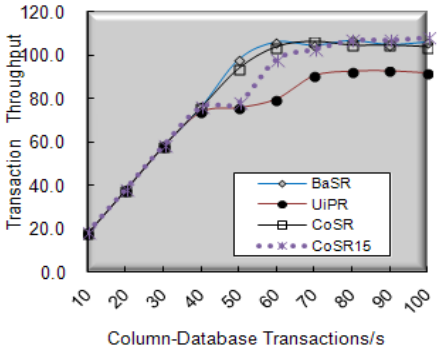

Fig. 2. Transaction Throughput

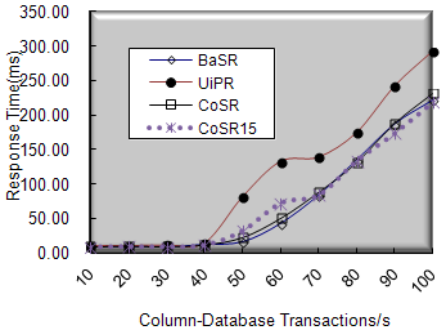

Fig. 3. Transaction Response Time

수치적으로 살펴보면, 초당 트랜잭션 발생량이 40~70개를 넘으면서 트랜잭션 처리치가 더 이상 증가하지 않고 서서히 처리 성능이 감퇴됨을 알 수 있다. 즉, 그 이상으로 시스템에 트랜잭션 처리를 요청하여도, 더 이상 시스템이 처리할 수 없고, 오히려 데이터 처리 경합이 발생하여 작업부하만 증가시킴을 의미한다.

위 그래프의 분석결과로 CoSR이 UiPR 대비 17% 더 높은 트랜잭션 처리치를 보였다. CoSR은 플래시 파일 시스템 내부에서 무효화되어서 폐기되는 데이터 페이지들을 소거 프로시저가 작동하기 전까지 최대한 쉐도우 페이지로 재활용함으로써 저장 공간 부담을 감소시킬 수 있다.

그림 4는 각 기법들의 저장 공간 사용량의 추이인데, 전술한 바와 같이 UiPR은 쉐도우 페이지 복구 기법처럼 이전 이미지를 저장하지 않으므로 저장 공간 사용량이 제일 적으며 큰 변화가 없다. 하지만, 전체 구간에서 CoSR가 BaSR을 비교해 보면, CoSR 기법이 훨씬 더 적은 양의 저장 공간을 사용하고 있다. 평균값으로 환산하면, CoSR가 BaSR에 비하여 39%정도로서 더 낮은 저장 공간을 사용하고 있다. 이는 쉐도우 페이지 저장 및 관리에 부하가 더 적음을 의미하며, 결과적으로 컬럼-지향 데이터베이스의 효율성을 증가시키게 된다. CoSR15는 CoSR 보다 더 적은양의 쉐도우 영역을 가지므로, CoSR 대비 54%정도의 저장 공간을 사용하고 있다. 반면에 CoSR15는 CoSR에 비하여 재활용 쉐도우 영역이 충분치 않게 설정하여 의도적으로 오버플로우가 많이 나오게 설정한 경우이므로, CoSR 보다는 트랜잭션 처리 성능은 더 낮게 나타나게 된다.

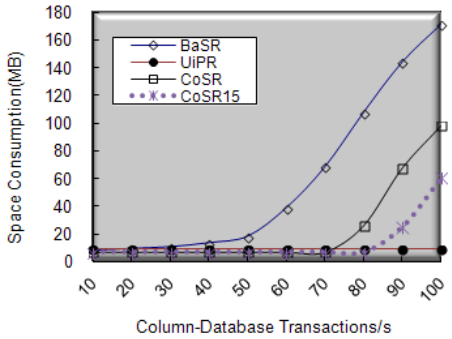

Fig. 4. Storage Space Consumption

5. 결 론

본 논문에서는 플래시 기반의 컬럼-지향 데이터베이스 환경에서 새로운 트랜잭션 회복 기법을 제안하였다. 먼저, 컬럼-지향 데이터베이스의 특성 및 플래시 메모리와의 연동, 그리고 플래시 메모리의 특성을 고려한 컬럼 기반의 데이터 처리를 위한 기반 구조 및 플래시 파일 시스템의 구조를 기술하였다.

제안된 컬럼 기반의 그림자 복구 기법은 즉시 갱신 복구 기법에 비하여 압축 부하가 감소시키고, 재활용 쉐도우 리스트 구조를 통하여 버려지는 데이터 페이지를 재차 이용함으로써 기존 쉐도우 페이징의 추가 저장 공간 부담을 완화시켜 컬럼-지향 데이터베이스 시스템의 저장 효율과 처리 성능을 높였다. 또한, 제안 기법의 효과를 검증하기 위하여, 윈도우 서버에서 대용량 실험 데이터를 구축하였고, 약 17% 더 높은 처리 성능을 얻었다.

References

[1] S. Ahn, K. Kim. "A Join Technique to Improve the Performance of Star Schema Queries in Column-Oriented Databases", Journal of Korean Institute of Information Scientist and Engineers, Vol. 40, No.3, pp. 209-218, 2013.6.

[2] S. Byun, Column-aware Transaction Management Scheme for Column-Oriented Databases, Journal of the Korean Society Internet Information, Vol. 15, No. 4 pp. 49-56, 2014
DOI: http://dx.doi.org/10.7472/jksii.2014.15.4.125

[3] D. Abadi, A. Boncz, and S. Harizopoulos, “Column-oriented Database Systems”, Proc. of the VLDB, Lyon, France, August 24-28 2009.
DOI: http://dx.doi.org/10.14778/1687553.1687625

[4] S. Harizopoulos, V. Liang, D. J. Abadi, and S. Madden, “Performance tradeoffs in read-optimized databases”, Proc. of VLDB, pp. 487 – 498, 2006.

[5] S. Byun. “Search Performance Improvement of Column-oriented Flash Storages using Segmented Compression Index”, Journal of the Korea Academia-Industrial, Vol. 14, No.1, pp. 393 – 401, 2013.
DOI: http://dx.doi.org/10.5762/KAIS.2013.14.1.393

[6] Lucas Mearian, “Analysis: SSD performance -- is a slowdown inevitable?”, Available From: http://www.computerworld.com/s/article/9132668/Analysis_SSD_performance_is_a_slowdown_inevitable_?taxonomyId=19& pageNumber=3, (accessed July, 2014)

[7] D. Abadi, S. Madden, and M. Ferreira. “Integrating compression and execution in column-oriented database systems”, Proc. of SIGMOD, pp. 671 – 682, 2006.
DOI: http://dx.doi.org/10.1145/1142473.1142548

[8] Tamer Ozsu, and Patrick Valduriez, Principles of Distributed Database Systems, Springer New York, 2011.

[9] Vijay Kummar, Albert Burger, "Performance Measurement of Main Memory Database Recovery Algorithms Based on Update-in-Place and Shadow Approaches", IEEE Transactions on Knowledge and Data Engineering, 4(6), 1992, pp. 567-571.
DOI: http://dx.doi.org/10.1109/69.180607

[10] Jack Kent, Hector Garcia-Molina, "Optimizing Shadow Recovery Algorithms", IEEE Transactions on Software Engineering, 14(2), Feb. 1988, pp. 155-168.
DOI: http://dx.doi.org/10.1109/32.4635

[11] J. Kim, S.Joo, H. Kang, An Efficient Recovery System for Spatial Main Memory DBMS, Journal of the Korea Spatial Information Society, Vol 8 No. 03, pp. 1-14, 2006.12

[12] E. M. Song, Y. K., Kim and C. H., Ryu "No-Log Recovery Mechanism Using Stable Memory for Real-Time Main Memory Database Systems", RTCSA'99, IEEE CS, Dec 1999, pp. 428-431.

[13] Chang L. and Kuo T., "An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems", in:Proc. 8th IEEE Real-Time and Embedded Technology Symposium, California, San Jose, 2002, pp. 187-196.
DOI: http://dx.doi.org/10.1109/RTTAS.2002.1137393

[14] Mendel Rosenblum John K. Ousterhout, "The design and implementation of a log-structured file system", ACM Transactions on Computer Systems, 10(1), February 1992, pp. 26-52.
DOI: http://dx.doi.org/10.1145/146941.146943

[15] JFFS,Available From: http://developer.axis.com/software/jffs/, (accessed Oct., 2014)

[16] Mesquite, CSIM2.0 Development Toolkit for Simulation and Modeling, Available From: http://www.Mesquite.com/documentation/documents/CSIM20_User_Guide-C.pdf, (accessed Dec., 10, 2014)

변 시 우(Siwoo Byun) **[정회원]**

- 1989년 2월 : 연세대학교 이과대학 전산과학과(공학사)
- 1991년 2월 : 한국과학기술원 전산학과(공학석사)
- 1999년 8월 : 한국과학기술원 전산학과(공학박사)
- 2000년 3월 ~ 현재 : 안양대학교 디지털미디어학부 정교수

<관심분야>
고속 데이터베이스, 대용량 저장장치, 스마트 시스템 등