데이터베이스 백업 성능 향상을 위한 병렬 인덱스 스캔 기법

이민욱, 한재현, 손용석
중앙대학교 컴퓨터공학부
makerdark98@cau.ac.kr, whffu2421@cau.ac.kr, sysganda@cau.ac.kr

Improving Performance of Database Backup via Parallel Index Scan

Min-uk Lee, Jaehyun Han, Yongseok Son*
School of Computer Science and Engineering, Chung-Ang University

요 약

데이터베이스 백업은 데이터베이스 데이터를 보존 및 복구하기 위해 사용되는 필수 기법이다. 이를 위해 데이터베이스 관리 시스템(Database Management System)에서는 백업 기능을 수행하는 응용을 제공한다. 하지만 이러한 백업 응용은 기존 HDD 의 특성을 고려하여 설계 및 구현되어있기 때문에 최근 주로 사용되고 있는 저장장치인 SSD 에서는 SSD 의 성능을 최대한 이끌어내지 못하고 있다. 예를 들어, 기존 백업 응용에서는 단일 스레드가 백업을 위한 데이터를 순차적으로 접근하고 풀 스캔 방식으로 백업을 수행한다. 따라서 본 연구에서는 SSD 의 병렬적 특성을 활용하여 백업 응용의 성능을 극대화시키는 것을 목표로 한다. 이를 위해 본 논문은 병렬 인덱스 스캔 기법을 제시한다. 제안한 기법은 기존 백업 기법과 달리 멀티 스레드를 활용하여 인덱스 스캔을 수행하고 병렬적으로 데이터에 접근하여 백업을 수행한다. 해당 기법은 기존 기법에 비해 최대 2.5 배 성능을 향상시켰다.

1. 서론

데이터베이스는 다양한 분야에서 널리 사용되고 있는 소프트웨어이며 시스템 오류 및 하드웨어 고장으로 인한 데이터베이스 데이터의 손실을 막기 위해서는 데이터베이스 백업 및 복구는 필수적이다[1]. 이러한 데이터 손실을 막기 위해서 데이터베이스 관리 시스템 (Database Management System)에서는 백업 기능을 수행하는 응용을 제공한다. 하지만 이러한 백업 응용은 기존 하드디스크 (HDD)의 특성을 고려하여 설계 및 구현되었기 때문에 최근 주로 사용되고 있는 고성능 저장장치 (SSD)에서는 기대만큼의 백업 성능을 보여주고 있지 못하고 있다.

본 논문에서는 고성능 저장장치 (SSD)의 병렬적 특성을 활용하여 백업 응용의 성능을 극대화시키는 것을 목표로 한다. 이를 위해 본 논문은 병렬 인덱스 스캔 기법을 제시한다. 해당 기법은 멀티 스레드 기법을 활용하여 스레드별로 스캔할 인덱스를 할당해준다. 그 후 각각의 스레드는 병렬적으로 인덱스 스캔을 수행하고 인덱스에 해당되는 백업에 접근한다. 특히, 해당 기법은 row 단위로 병렬화를 수행하기 때문에 하나의 테이블에서도 효용성이 있다.

본 논문은 해당 기법을 구현하기 위해 가장 널리 사용되고 있는 MySQL 의 mysqldump[2]를 활용하였으며, 해당 기법은 기존 기법 대비 최대 2.5 배 성능을 향상시켰다. 본 논문의 나머지 부분은 다음과 같이 구성된다. 2 장은 본 논문과 관련된 배경에 대해 논하고, 3 장에서는 제안된 기법을 설명하고 4 장에서는 실험결과를 보이고, 5 장에서는 이를 통한 결론을 맺는다.

Acknowledgement
본 연구는과학기술정보통신부 및 정보통신기획평가원의 2021 년도 SW 중심대학지원사업의 연구결과로 수행되었음 (2017000100005 1001)
(교신저자: 손용석)

2. 배경

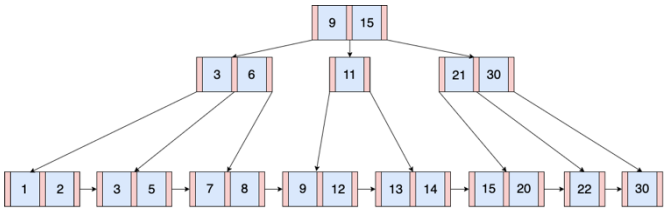
2.1 저장매체

저장매체의 속도가 컴퓨터 매체 중 가장 느리기 때문에 대부분의 컴퓨터 시스템들의 병목은 저장매체에 있었다 [3]. 또한 이러한 병목을 해소하고자 HDD 기반 응용들은 HDD 특성을 파악하여

순차적인 접근을 통한 성능 향상을 목표로 하였다. 하지만 최근 주로 사용되고 있는 SSD 의 경우에는 여러 채널들을 통해 병렬적으로 접근이 가능하여 병렬적으로 입출력을 수행했을 때 성능이 극대화된다. 따라서 이러한 SSD 의 병렬성을 최대로 활용하는 것이 컴퓨터 시스템의 성능을 극대화시킬 수 있는 방법이다 [4][5]. 본 논문은 병렬성이 높은 SSD 기반에서 데이터베이스 백업 성능을 극대화시키기 위한 기법을 제시한다.

2.2 B+ tree 와 인덱스 스캔

B+ tree 는 데이터베이스 시스템에 널리 쓰이는 자료구조로 트리를 구성하는 모든 노드(node)들은 키와 링크를 가지고 있으며, 최하단 노드를 제외한 노드(index node)들은 링크들이 자식 노드를 가리키며, 최하단 노드(leaf node)들은 노드의 마지막 링크가 다음 최하단 노드를 가리키도록 구성되어 있다 [6]. 이는 B-Tree 를 확장하여 자식 노드의 마지막 링크를 동일한 트리 레벨의 다음 노드를 가리키게 함으로써 데이터를 특정 인덱스의 값부터 순차적으로 읽는 연산에 적합한 자료구조이다. 그림 1 은 B+-Tree 의 개략적인 모습을 보여준다.



(그림 1) B+-Tree 자료구조

그림 1 에서의 각 노드의 내부 숫자는 키를 나타내며, 화살표는 링크를 나타낸다. 이러한 자료구조를 활용해서 데이터베이스 시스템은 일부 노드에만 접근하여 조건 질의를 빠른 속도로 처리한다. 이렇게 특정한 키를 통한 질의문을 빠르게 처리하는 방식을 인덱스 스캔이라한다 [7].

2.4 데이터베이스 덤퍼(Database Dumper)

데이터베이스 시스템은 데이터베이스의 복제 및 장애 대처를 위한 백업 기능을 제공하기 위하여 백업 응용을 지원한다. 최근 가장 널리 사용되고 있는 데이터베이스 중 하나인 MySQL 은 mysqldump 라는 백업 응용을 제공한다. 해당 응용은 데이터베이스 시스템의 서버, 클라이언트 구조를 유지하며 클라이언트 소프트웨어로서 동작한다[8]. 즉, 해당 응용은 질의문을 데이터베이스 시스템에 요청하고, 이에 대한 응답을 통해서 새롭게 데이터베이스를 구성할 수 있는 삽입문을 만들어낸다. 하지만, 이는 단일 스레드로 동작하고 데이터베이스의 파일을 인덱스 사용 없이 순차적으로 읽는 풀 스캔을 수행한다. 이는 SSD 의 병렬성과 질의문을 빠르게 처리하기 위해 구성된 구조를 활용하지 못하고 있다.

3. 제안된 기법

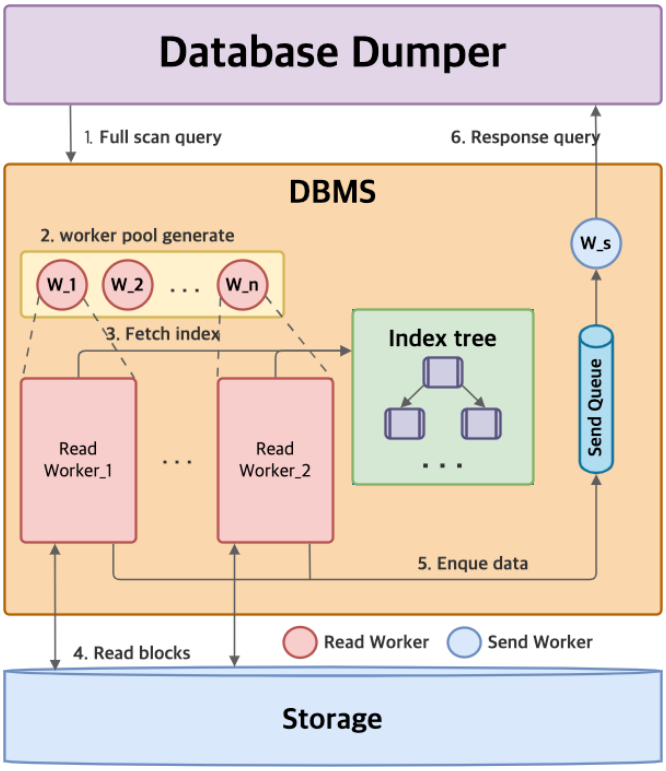
3.1 기본 아이디어

2.1 절에서 언급하였듯이, 병렬성을 제공하는 SSD 와 같은 저장장치에서는 단일 스레드로 순차적 접근을 수행하는 풀 스캔은 최적의 성능을 이끌어내지 못한다. 따라서 본 논문에서는 풀 스캔이 아닌 인덱스 스캔을 통한 멀티 스레드 기법을 제시한다.

3.2 설계 및 구현

데이터베이스 시스템은 서버-클라이언트 구조로 인해서, 백업 응용은 백업하고자 하는 대상 테이블의 인덱스 구조에 대해서 알 수 없다. 또한 만약 이러한 구조를 파악하더라도 여러 개의 질의문을 동시에 처리하기 위해서 다중 연결을 해야하는 문제점이 있다. 따라서 클라이언트의 변경에는 한계가 있어 본 논문은 데이터베이스 시스템 자체의 풀 스캔처리 로직을 병렬적으로 설계 및 구현한다.

이를 위해 본 논문에서는 풀 스캔 처리에 대해서 테이블 데이터 파일 전체를 순차적으로 읽는 것이 아니라 인덱스 스캔을 활용하여 병렬 처리하는 방법을 제시한다. 또한 본 논문은 데이터를 병렬적으로 접근하는 접근 스레드와 백업 응용과 데이터베이스 서버 연결을 유지하며 데이터를 보내주는 서버에서 응용으로 전송하는 스레드로 나누어서 백업 처리를 수행한다.



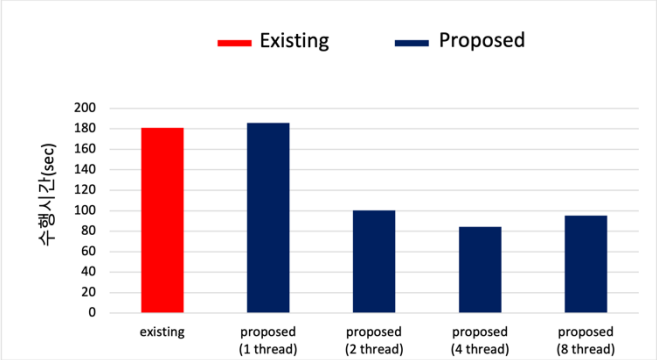
(그림 2) 본 논문에서 제시하는 멀티 스레드를 통한 데이터베이스 백업

그림 2 는 본 논문에서 제시하는 멀티 스레드를 통한 데이터베이스 백업 기법을 나타낸다. 그림과 같이 백업 응용인 덤퍼는 테이블 풀 스캔을

데이터베이스 시스템에 요청한다 (①). 그 후 요청을 받은 데이터베이스 시스템은 접근 스레드풀과 전송 스레드를 구성한다 (②). 접근 스레드 (worker)는 인덱스 트리에서 각자 처리해야 할 인덱스를 읽어온다 (③). 접근 스레드는 각 인덱스에 해당하는 데이터를 읽고 처리한다 (④). 그 후 접근 스레드는 읽어온 데이터를 전송할 데이터 형태로 구성한 후 전송 큐에 넣어준다 (⑤). 마지막으로 전송 스레드는 전송큐에서 데이터를 읽어온 후 해당 데이터를 응용으로 전송한다 (⑥).

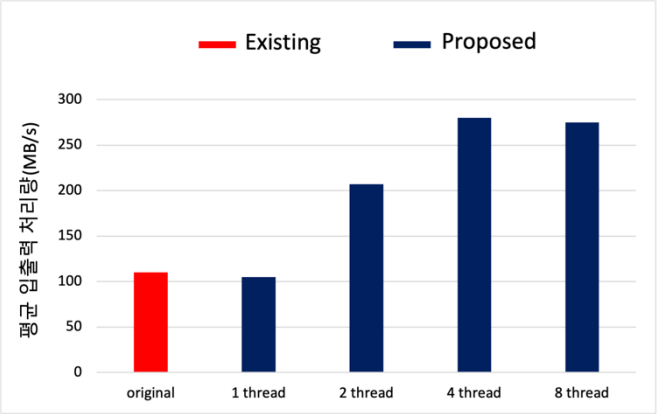
4. 실험 결과 및 분석

본 논문은 실험 평가를 위해 Intel® Xeon® CPU E7-8867 v4 @ 2.4GHz (18core * 4), 16G Memory, Intel P3700 를 구성하였다. 또한 백업데이터를 구성하기 위해서 sysbench 를 사용하여 데이터 (30G) 생성한 후 백업(덤프)을 수행하였다.



(그림 3) 기존 응용과 제안한 응용의 백업 수행시간

그림 3 는 기존 백업 응용과 제안한 백업 응용의 백업 수행 시간을 보여준다. 그림과 같이 1 스레드에서는 제안된 기법은 인덱스를 읽는 연산이 추가적으로 발생하기 때문에 기존 응용에 비해 성능이 3.8%만큼 하락한다. 제안된 응용은 기존 응용에 비해 4 스레드 일 때 최대 성능을 보여주며 2.1 배의 성능을 보인다.



(그림 4) 기존 응용과 제안한 응용의 백업 입출력 처리량

그림 4 은 기존 백업 응용과 제안한 백업 응용에서 백업 수행 시 발생되는 입출력 처리량을 보여준다. 그림과 같이 제안한 백업 응용에서 스레드 개수가 1 개일 때는 인덱스를 읽는 오버헤드로 인하여 입출력 처리량이 감소한다. 반면에 스레드 개수가 증가함에 따라 제안한 응용은 기존 응용 대비 입출력 처리량이 증가하며 스레드 개수가 4 개일 때 최대 2.5 배만큼 성능이 증가하였다.

5. 결론 및 향후 연구 계획

본 연구에서는 SSD 의 병렬적 특성을 활용하여 백업 응용의 성능을 극대화시키기 위해 병렬 인덱스 스캔 기법을 제시하였다. 제안한 기법은 기존 백업 기법과 달리 멀티 스레드를 활용하여 인덱스 스캔을 수행하고 병렬적으로 데이터에 접근하여 백업을 수행한다. 이러한 제안한 기법은 기존 기법 대비 최대 2.5 배 성능을 향상시켰다.

향후 연구로는 스레드 수의 증가에 따른 확장성 문제를 해결할 계획이다. 해당 문제는 데이터베이스 시스템의 버퍼 관리자 오버헤드 및 동시성을 감소시키는 락킹 (locking) 문제 때문에 발생되고 있다. 따라서 데이터베이스 시스템 내부 구조를 추가적으로 연구하여 스레드 수의 증가에 따른 확장성을 향상시킬 계획이다.

참고문헌

[1] S. Bhattacharya et al. Coordinating backup/recovery and data consistency between database and file systems. In SIGMOD, pages 500–511, New York, NY, USA, 2002. ACM.

[2] (2005) mysql, mysqladmin, and mysqldump. In: The Definitive Guide to MySQL5 Apress.

[3] D. DeWitt. From 1 to 1000 MIPS. PASS Summit 2009 Keynotes(The Professional Association for SQL Server), November 2009

[4] Eshghi K., Micheloni R. (2013) SSD Architecture and PCI Express Interface. In: Inside Solid State Drives (SSDs). Springer Series in Advanced Microelectronics, vol 37. Springer, Dordrecht.

[5] S.-W. Lee, B. Moon, and C. Park. Advances in Flash Memory SSD Technology for Enterprise Database Applications. In Proceedings of SIGMOD, pages 863–870, 2009.

[6] Navathe, Ramez Elmasri, Shamkant B. (2010). Fundamentals of database systems (6th ed.). Upper Saddle River, N.J.: Pearson Education. pp. 652–660

[7] Oracle. Optimizer Access Paths https://docs.oracle.com/database/121/TGSQL/tgsql_optop.htm 2013

[8] DJ DeWitt, N Kabra, J Luo, JM Patel, JB Yu. Client-Server Paradise VLDB, 1994