Spark 기반의 인 메모리 분산 이동객체 색인 기법

In-memory Distributed Moving Object Indexing Method based on Spark

이 윤 수, 송 석 일
한국교통대학교 컴퓨터공학과

Yunsou Lee, Seokil Song
Department of Computer Engineering, Korea National University of Transportation

요약

이 논문에서는 다수 노드로 구성되는 클러스터 환경을 위한 인-메모리 이동객체 분산 색인기법을 제안한다. 제안하는 방법은 Spark Stream의 D-strem모델을 사용하여 처리율 저하등의 문제를 유할 할 수 있는 잠금 기반의 동시성 제어방법을 배제한다.

I. 서론

최근 몇 년 동안에 스마트폰, 태블릿 PC와 같은 모바일 기기가 대중화가 가속되면서 GPS기술과 무선 통신 기술을 기반으로 모바일기기의 위치 측정 및 활용이 가능해졌다. 이와 함께 모바일 위치기반서비스가 주요한 인터넷 응용 중의 하나가 되었다. 모바일 기기의 수가 증가하면서 서버 측에서 처리해야 할 일이 증가하였다. 특히, 대량의 이동객체에 대한 위치관련 질의를 처리하고, 이동객체의 위치 변경을 처리하는 일이 매우 중요해졌다.

일반적인 위치기반 응용에서 이동객체는 일정주기마다 자신의 위치를 서버로 전송한다. 높은 이동객체의 위치 정확도가 요구되는 응용에서는 전송주기가 짧아져야 한다. 따라서, 이동객체의 수가 매우 많거나 높은 정확도가 요구되는 응용의 경우 서버의 부하는 매우 커진다. 예를 들어 이동객체가 6,000,000개이고 전송주기가 1분일 경우 서버는 10마이크로초 마다 1번씩 업데이트를 수행해야 하므로 1초에 10,000번의 업데이트를 수행해야 한다.

이와 같은 서버 작업 부하는 디스크기반 데이터구조와 알고리즘으로 처리하는 것이 불가능해진다. 이를 해결하기 위해 일부 인-메모리 기반 이동객체 색인기법이 제안된 바 있다. [1-4]. [1] 에서는 R-트리를 CPU캐시를 고려하여 최적화하고 있다. MOVIES[3]는 메인 메모리에 색인 구조를 일시적으로 유지하고 버리는 형태의 기법을 제안하고 있다.

또한[4]에서는 다중코어 프로세서의 병렬성을 이용할수 있는 메인 메모리 색인기법이 제안되었다. 이 기법에서는 일관성 있는 데이터베이스 상태를 유지하고 정확한 질의 결과를 반환하기 위해 기존의 색인기술과 잠금 기술을 이용하여 업데이트와 질의간의 충돌을 회피하고 있다. 이를 위해 적은 데이터를 짧은 시간만 잠금을 걸어서 무거운 잠금의 부하를 피하도록 하는 경량의 잠금 기술이 사용된다.

이 논문에서는 잠금 기반의 동시성 제어방법은 스레싱(Thrashing)과 같은 몇 가지 문제가 항상 발생할 수 있으므로 사용을 최소화 할 필요가 있다는 결론에 도달했다. 또한 다중 코어 프로세서 시스템뿐만 아니라 다중 노드의 병렬 처리를 통해 처리율을 높일 수 있다는 것을 전제로 제안하는 기법을 설계 하였다. 이 논문에서는 다중 노드의 병렬처리를 이용한 분산 이동객체 색인기법을 제안한다. 제안하는 기법은 처리량을 증가시키는 잠금 기술을 기반으로 하지 않는다.

제안하는 색인 기법은 Spark Stream[5]을 기반으로 한다. Spark Stream에서는 Discretized streams (D-Streams) 모델을 제안하고 이를 기반으로 스트림을 처리한다. D-Streams 모델에서는 입력되는 데이터 스트림을 1초 이하의 시간동안 모아서 배치처리를 수행한다. 이때의 시간 주기는 보통 0.5초 내외이다. 각 배치처리가 수행되는 데이터 집합은 Spark에서 제안하는 RDD (Relilient Distributed Dataset) 모델에 의해 내결함성의 성질을 갖게 된다. 따라서, 인 메모리의 데이터 스트림을 잃게 되더라도 병렬로 복구를 수행할 수 있어 효율적인 복구가 가능하다.

Ⅱ. 제안하는 분산 이동객체 색인기법

제안하는 분산 색인 기법은 D-stream 모델을 기반으로 한다. D-stream은 사용자가 응용 프로그램을 구축 할 수 있도록 두 가지 유형의 연산을 제공한다. 첫 번째로 하나 또는 그 이상의 상위 스트림으로부터 새로운 D-stream을 생성하는 변환연산자가 있다. 두 번째 로는 HDFS과 같은 외부시스템에 응용 프로그램의 데이터를 출력하는 출력 연산자가 있다. D-stream은 map, reduce,

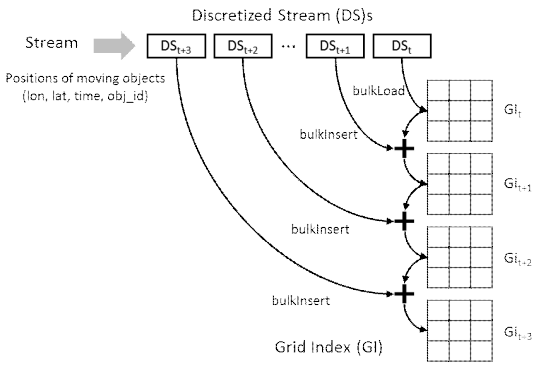
groupBy, 그리고 join과 같은 Spark에서 사용가능한 비 상태유지 변환연산자를 제공한다.

제안하는 색인기법은 이동객체 색인과 질의를 위해 Spark Stream에서 몇 가지 변환연산자와 출력연산자를 추가한다. 표1은 제안하는 새로운 연산자를 나타낸다. 제안하는 연산자는 3개의 변환연산자와 하나의 출력연산자로 구성된다. DS_t는 시간t 시점의 D-stream을 의미하고 GI_t는 시간t 시점의 그리드 인덱스를 의미한다.

표 1. 제안하는 색인기법의 연산자

Operators	Function
bulkLoad (transform)	Build an initial index Transform DStintoGIt
bulkInsert (transform)	Insert a set of position data into an index Transform GItwithDSt+1intoGIt+1
splitIndex (transform)	Split a index into two according to time Transform GItintoGIt_upperandGIt_lower
search (output)	Operators for querying such as range queries and KNN queries

입력 스트림은 주기적으로 전송되는 이동객체의 위치이다. Spark Stream은 입력 스트림을 D-stream으로 변환한다. 그림1에서와 같이 입력 스트림은 Spark Stream에 의해 DS_t,DS_{t+1},DS_{t+2}, 그리고 DS_{t+3}로 변환된다. 제안하는 색인방법은 각각의 D-stream에 bulkLoad 및 bulkInsert 연산을 수행한다. bulkLoad는 DS_t에서 위치 데이터와 그리드 인덱스 GI_t을 생성한다. 제안하는 색인방법에서 인덱스의 생성과 업데이트에 소요시간을 감소시키기 위해 Grid Index를 사용한다.



▶▶ 그림 1. 입력스트림으로 생성되는 Grid 인덱스

제안하는 색인기법은 동시성제어를 위한 잠금 기반 제어기법을 사용하지 않는다. Spark Stream의 D-stream모델은 불변이므로 갱신과 탐색은 인덱스에서 동시에 수행되지 않는다. 그림1과 같이 인덱스가 bulkLoad 및 bulkInsert연산자에 의해 갱신되고 사용자가 접근할 수 있는 시간 t, t+1, t+2, 그리고 t+3의 다중버전의 인덱스는 메인 메모리에 존재한다. 그러므로 GI_{t+3}이 구축되는 동안 사용자는 GI_{t+2}에 접근하게 된다.

Ⅲ. 결론

제안하는 분산 이동객체 색인기법은 Spark Stream을 기반으로 한다. 제안하는 방법은 bulkLoad, bulkInsert, splitIndex, 그리고 search 연산자로 구성된다. bulkLoad, bulkInsert 그리고 splitIndex는 변환연산자로서 새로운 RDD를 생성하고 search연산은 출력연산자로서 조회결과를 반환한다. 향후연구로는 Spark Stream을 기반으로 하여 제안하는 색인기법을 구현하고 성능평가를 수행 할 예정이다.

■ 참 고 문 헌 ■

[1] K. Kim, S. K. Cha, and K. Kwon. Optimizing multidimensional index trees for main memory access. SIGMOD Rec., 30(2):139–150, 2001.

[2] L. Biveinis, S. ˇSaltenis, and C. S. Jensen. Main-memory operation buffering for efficient r-tree update. In VLDB, pages 591–602, 2007.

[3] J. Dittrich, L. Blunschi, and M. A. V. Salles. Indexing moving objects using short-lived throwaway indexes. In SSTD, pages 189–207, 2009.

[4] Šidlauskas, Darius, Simonas Šaltenis, and Christian S. Jensen. "Parallel main-memory indexing for moving-object query and update workloads." Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012.

[5] Zaharia, Matei, et al. "Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters." Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing. USENIX Association, 2012.