0907

**논문 선정 배경**

**Spark에서 공간 쿼리를 지원하는 효과적인 프레임 워크 GeoSpark SQL**

**GeoSpark SQL - An Effective Framework Enabling Spatial Queries on Spark**

빅 데이터 시대에는 다양한 위치 기반 서비스와 같은 인터넷 기반 공간 정보 서비스가 모든 곳에 배치되고 그에 따라 방대한 공간 데이터에 대한 쿼리가 증가하고 있는 반면, 기존의 관계형 데이터베이스(데이터를 테이블 형태로 저장, 항목=행 / 항목의 속성=열)는 대규모 공간 쿼리 처리의 요구에 대응하기가 어렵다.

**Spark sql**(HIVE가 hadoop에 저장된(HDFS) 데이터를 쿼리로 호출)은 **공간 쿼리에 직접 사용이 불가능**함. 따라서 spark sql 확장을 통해 문제를 해결해야 한다. **첫째**, spark sql을 포함한 spark는 메모리 컴퓨팅 프레임 워크일뿐이므로 공간 인덱스 뿐만 아니라 적절한 공간 데이터 외부 저장 방법을 설계해야 한다. **둘째**, spark sql은 공간 함수 또는 작업을 지원하지 않으므로 공간 질의 요건을 충족시키기 위해 spark 환경의 공간 연산자를 확장해주어야 한다. **마지막**으로, spark sql이 제공하는 내부 쿼리 최적화 도구 외에도, 외부 공간 쿼리 최적화 도구를 설계하여 공간 쿼리를 최적화 시킴으로써 성능을 향상 시킨다.

점점 더 커지는 공간 쿼리 처리 요구 사항을 해결하기 위해 분산 컴퓨팅 **프레임워크**인 Spark(빅데이터를 처리하기 위해 오픈소스로 생성 된 병렬분산처리 플랫폼)를 사용해야 하는데, Spark에서 지리적 데이터 처리(공간 쿼리)를 가능하게 하는 geoSpark SQL 프레임워크가 있다. 이는 Hive와 Spark를 통합하여 효율적인 스토리지 관리와 고성능 병렬 컴퓨팅을 가능하게 해주기 때문에 방대한 지리적 데이터를 처리하는 데에 있어 많은 편리성을 제공한다.

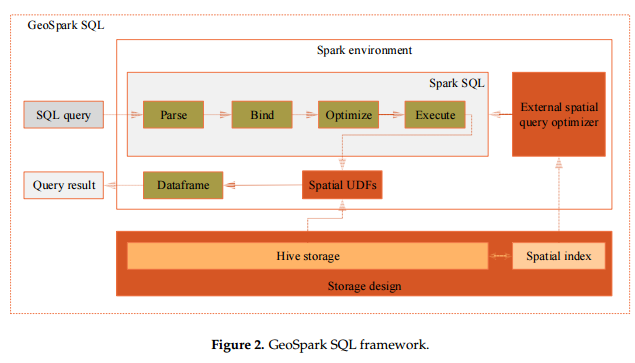
본 논문에서는 GeoSpark SQL 프레임 워크에 따른 스토리지 관리 방법, Spark 실험적 평가에 따른 공간 쿼리 최적화 방법 등과 같은 내용을 다루고 있기 때문에 많은 참고가 될 것 같아 선정하게 되었다.

***논문으로 도출하고자 하는 것***

* 대중교통(버스, 지하철) 좌표 및 승하차 그리고 서울시 구 단위 경계 데이터를 사용하여 승객들의 대중교통 이용 및 구 단위로 위치한 버스 정류소 및 지하철 역 등에 대해 분석
  + - Spark 와 Geospark 사용
    - 분석은 Join 연산 사용
    - Geospark에선 최종적으로 다양한 map을 사용해 결과를 시각적으로 보일 생각
* 분석을 하면서 이에 사용되는 spark와 geospark에 대한 성능을 비교하고 분석

0913

참고 논문 블록도



GeoSpark SQL FLOW 설명

공간데이터는 Hive와 같은 외부 저장소에 저장되며, **공간 데이터를 필요로 하는 요청을 받아 공간데이터를 전송하게 되었을 때, Geospark SQL은 공간 데이터를 메모리로 읽어 공간 연산을 수행하고 요청 결과**를 얻는다. (블록도 주제)

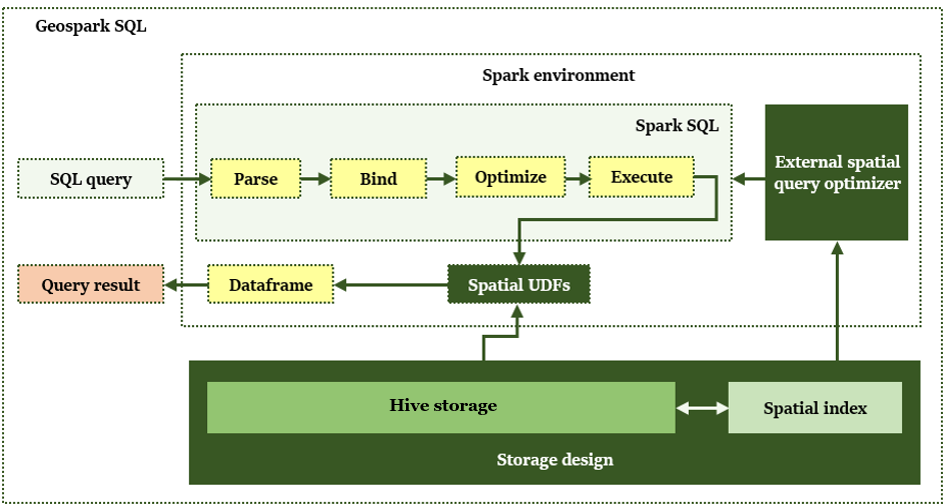
SQL문은 spark sql에 내장된 sql parse에 의해 파싱되어 구문 트리를 구축한다. 이 과정에서 SELECT, FROM, WHERE, 표현식 등과 같이 sql에서 사용되는 핵심 단어들의 경우 모두 분리되어 있기 때문에 구문에 대한 공간 연산의 의미를 이해하고 실행할 수 있다.

SQL문은 공간 데이터 스키마(컬럼, 테이블, 뷰 등)를 포함하고 있어야 하며, 해당 과정은 Spark SQL 에서도 수행된다.

Spark SQL을 통해 전처리를 거치게 되면, **내부 쿼리 최적화기**를 통해 쿼리의 성능을 높여 주어 최적의 쿼리를 실행하게 되며, 동시에 **외부 공간 쿼리 최적화**기는 공간 인덱스를 통해 I/O 비용을 절감하여 물리적 공간 쿼리를 더욱 최적화 시킨다.

마지막으로, 실행 단계에서는 확장 공간 UDF(사용자 정의 함수)의 지원을 받아 “데이터에 접근->데이터 액세스(등록,수정,삭제,검색) 및 저장->결과”의 순서로 물리적 질의 계획을 수행한다. 그리고 메모리에서 공간 쿼리의 결과는 DataFrame 객체의 형태로 나타난다. (df는 spark의 일반적인 RDD(스파크 데이터 분석의 기본적 단위) 기반 분산 데이터 세트이다.)

내 블록도



**공간 인덱스**는 R-Tree 인덱스 알고리즘을 이용해 2차원의 데이터를 인덱싱하고 검색하는 목적의 인덱스

[파티셔닝 정보](https://m.blog.naver.com/syung1104/221103154997" \o "https://m.blog.naver.com/syung1104/221103154997)

[파티션 재구성 = 각각의 구단위 경계 데이터를 기준으로 해당 범위 안에 있는 지리 데이터를 동일한 노드로 모아 재구성](https://m.blog.naver.com/syung1104/221103154997" \o "https://m.blog.naver.com/syung1104/221103154997)

[인덱싱 : 파티션 작업이 끝난 데이터들 중 같은 정보를 가지고 있는 데이터들 끼리 묶어서 인덱스 생성](https://m.blog.naver.com/syung1104/221103154997" \o "https://m.blog.naver.com/syung1104/221103154997)