KIPS Tr. Software and Data Eng.
Vol.5, No.10 pp.477~482 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2016.5.10.477

Design of Spark SQL Based Framework for Advanced Analytics

Jaehwa Chung†

ABSTRACT

As being the advanced analytics indispensable on big data for agile decision-making and tactical planning in enterprises, distributed processing platforms, such as Hadoop and Spark which distribute and handle the large volume of data on multiple nodes, receive great attention in the field. In Spark platform stack, Spark SQL unveiled recently to make Spark able to support distributed processing framework based on SQL. However, Spark SQL cannot effectively handle advanced analytics that involves machine learning and graph processing in terms of iterative tasks and task allocations. Motivated by these issues, this paper proposes the design of SQL-based big data optimal processing engine and processing framework to support advanced analytics in Spark environments. Big data optimal processing engines copes with complex SQL queries that involves multiple parameters and join, aggregation and sorting operations in distributed/parallel manner and the proposing framework optimizes machine learning process in terms of relational operations.

Keywords : Dvanced Analytics, Spark, Complex Query, Distributed Processing Platform, Hadoop, Mapreduce

Spark SQL 기반 고도 분석 지원 프레임워크 설계

정 재 화†

요 약

기업의 신속한 의사결정 및 전략적 정책 결정을 위해 빅데이터에 대한 고도 분석이 필수적으로 요구됨에 따라 대량의 데이터를 복수의 노드에 분산하여 처리하는 하둡 또는 스파크와 같은 분산 처리 플랫폼이 주목을 받고 있다. 최근 공개된 Spark SQL은 Spark 환경에서 SQL 기반의 분산 처리 기법을 지원하고 있으나, 기계학습이나 그래프 처리와 같은 반복적 처리가 요구되는 고도 분석 분야에서는 효율적 처리가 불가능한 문제가 있다. 따라서 본 논문은 이러한 문제점을 바탕으로 Spark 환경에서 고도 분석 지원을 위한 SQL 기반의 빅데이터 최적처리 엔진 설계와 처리 프레임워크를 제안한다. 복수의 조건과 다수의 조인, 집계, 소팅 연산이 필요한 복합 SQL 질의를 분산/병행적으로 처리할 수 있는 최적화 엔진과 관계형 연산을 지원하는 기계학습 최적화하기 위한 프레임워크를 설계한다.

키워드 : 고도 분석, 스파크, 복합 질의, 분산처리 플랫폼, 하둡, 맵리듀스

1. 서 론

최근 빅데이터를 빠르고 정확하게 고도 분석(advanced analytics)할 수 있는 기술은 새로운 정보 및 지식 발견과 직·간접적으로 연결되어 있어 기업 및 국가 경쟁력의 차별화 전략 수립에 많은 주목을 받고 있다. 고도 분석이란 대용량의 데이터에서 의사결정을 지원하기 위해 축적된 데이터 안에서 숨겨져 있는 유용한 정보를 발견하기 위한 분석 기법으로 연관성 분석, 클러스터링, 분류 등 데이터 마이닝 기법에 기반한 기술 분석(descriptive analytics) 뿐만 아니라 미래에 발생할 사건의 확률이나 경향에 대한 예측을 수행하기 위한 의사결정나무, 회귀분석, 신경망, 유전자 알고리즘, 기계학습 등 예측분석(predictive analytics)을 의미한다. 빅데이터의 고도 분석을 기반으로 재난, 재해, 식품안전, 테러, 범죄 등 사회현안에 합리적·선제적 대응을 강화할 수 있다. 따라서 빅데이터 분석을 필요한 정보를 수집 및 분류하고 이를 분석하여 빠른 의사결정에 활용하는 기법에 대한 연구가 활발하게 진행되고 있다.

고도 분석 지원을 위한 시스템의 진화 과정은 Fig. 1과 같이 하둡 플랫폼 기반, 자체 엔진을 내장한 하둡 플랫폼 기반 그리고 스파크 플랫폼 기반으로 구분할 수 있다.

빅데이터 분석을 위해 하둡 플랫폼 기반의 SQL on Hadoop이 가장 우선적으로 등장하였다. 하둡(hadoop)은 빅데이터 처리를 위한 표준 플랫폼으로 하둡 분산 파일 시스템(HDFS)과 데이터를 분산시켜 처리한 뒤 하나로 합치는

※ 본 논문은 2015학년도 후기 한국방송통신대학교 학술연구비 지원으로 수행되었음.
† 종신회원 : 한국방송통신대학교 컴퓨터과학과 조교수
Manuscript Received : July 22, 2016
Accepted : August 3, 2016
* Corresponding Author : Jaehwa Chung(jaehwachung@knou.ac.kr)

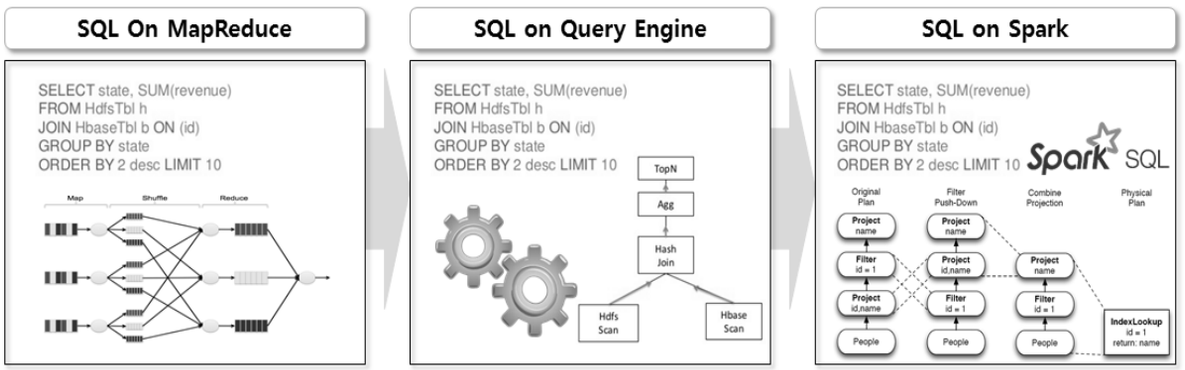

Fig. 1. The History of Distributed System for Supporting Advanced Analytics

기술인 맵리듀스(Map/Reduce)[1]를 구현한 자바 기반의 오픈소스 프레임워크에 SQL과 같은 고차원 언어를 지원하는 Pig[2], Hive[3] 등의 SQL on Hadoop 기술이 사용되었다.

SQL on Hadoop은 HDFS의 데이터에 SQL로 처리를 요청하고 분산 처리하는 시스템으로 데이터 플랫폼의 차세대 핵심 기술로 주목받고 있으나 맵리듀스는 관계형 처리가 아닌 맵과 리듀스 함수의 단일 입력과 단일 출력 형태의 알고리즘을 병렬적으로 처리하기 위한 목적으로 고안되어 관계 연산처리 및 복잡한 분석 알고리즘들을 처리하는데 한계가 있다.

초기 SQL on Hadoop의 한계를 극복하고자 탈 맵리듀스 컴퓨팅 모델(관계형 질의처리 엔진)을 탑재한 Tajo[4], Impala[5] 등의 시스템이 제안되었으나, 분석 대상 데이터가 디스크에 위치하고 중간 데이터는 메모리와 디스크를 동시에 사용하기 때문에 디스크 I/O가 많이 발생하고 중간데이터가 다른 노드로 복제되면서 네트워크 I/O 또한 크게 발생하는 문제가 있다. 따라서 분석 대상 데이터의 인메모리 처리를 기본으로 하여 보다 빠르고 지연 속도가 낮은 분석이 가능한 Spark[6]가 새로운 플랫폼이 대안으로 대두되었다.

SparkSQL[7]은 인메모리 분산 컴퓨팅 기술을 탑재한 Spark 플랫폼에 관계형 연산을 지원하는 데이터 모델인 DataFrame과 질의처리 엔진을 지원한다. 특히 Spark SQL은 확장 가능한 Catalyst 최적화기를 통해 분석적 SQL 질의를 최적화하며 질의 결과를 RDD(Resilient Distributed Datasets)로 변환하여 기존 Spark 상에서 기계학습과 같은 고도 분석을 지원한다.

데이터 분석을 통해 의사 결정을 내리기 위해서는 데이터의 단순 통계 정보나 단순 쿼리가 아닌 다수의 조인과 집계, 정렬 등이 포함된 Complex SQL 질의와 기계학습 등 계산 비용이 높은 데이터 처리가 요구되고 있으나 SQL on Hadoop과 Spark SQL은 Complex SQL 질의처리와 기계학습 연동처리가 요구되는 고도 분석을 위한 최적화가 어렵다.

따라서 본 논문에서는 조건 연산이나 Complex SQL 질의처리를 위한 분산/병행 최적화 기법 등의 연구를 통해 데이터 처리를 최소화 화기 위해 Spark SQL에서 고도 분석 지원을 위한 최적화 엔진을 설계하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서 SQL 기반의 빅데이터 처리 기술에 대한 연구 동향을 기술하고 3장에서는 고도 분석 지원을 위한 빅데이터 최적처리 프레임워크 설계한다. 마지막으로 4장에서는 본 논문에 대한 결론을 기술한다.

2. 관련 연구

이번 장에서는 SQL 기반 빅데이터 처리 기술에 대한 연구동향을 맵리듀스 기반의 SQL on Hadoop, 자체 질의처리 엔진 기반의 SQL on Hadoop과 Spark 기반의 SQL on Spark로 분류하여 설명한다.

2.1 맵리듀스 기반 SQL on Hadoop

맵리듀스 기반 SQL on Hadoop은 HDFS에 저장된 데이터 처리를 위해 맵리듀스 함수를 직접 프로그래밍하지 않고 SQL을 이용하여 데이터를 분산 처리하는 기술로, SQL을 맵리듀스로 변환하는 기술이다.

Hive는 HDFS에 분산·저장된 데이터를 읽어 맵리듀스 프레임워크를 이용해 분산 처리하는 데이터웨어하우징 솔루션으로 하둡 기반 질의 변환 엔진으로 자바를 이용하여 맵리듀스를 작성하지 않고 SQL만으로 하둡의 대용량 데이터 분석이 가능하다. 또한 SQL과 유사한 언어인 HiveQL을 이용해 질의를 요청하고 파서를 통해 명령을 분석한 후 해당 명령을 맵리듀스 형태로 변환하여 처리한다.

Pig는 고수준의 데이터 처리 구조를 제공하며, 이를 조합해서 대량의 데이터 처리를 가능하게 한다. SQL을 지원하기 위해 유사 형태의 Pig Latin이라는 스크립트 언어를 사용한다. Pig Latin은 절차적 프로그램 언어 형태로 대용량 데이터 처리 프로그램을 작성할 수 있다. Pig Latin로 작성된 SQL은 논리적인 실행 계획으로 변환되고, 다시 맵리듀스로 변환되어서 실행된다.

그러나 하둡은 자바 기반의 분산·병행 프레임워크로 전문 개발자가 아닌 일반 사용자들은 맵리듀스 프로그램을 작성하기 어렵다. 또한 Hive와 Pig의 경우 SQL을 내부적으로 맵리듀스로 변환하여 실행하는 방식이기 때문에 고성능으로 동작하기 어려우며 표준 SQL을 사용할 수 없다는 단점이 있다.

2.2 자체 질의처리 엔진 기반 SQL on Hadoop

자체 엔진 기반 SQL on Hadoop은 SQL을 시스템에서 개발한 질의처리 엔진으로 분석한 논리적/물리적 실행 계획을 분산 노드에서 실행하는 기술이다. SQL 실행을 맵리듀스와 같은 범용 분산 처리 시스템에 의존하지 않고 자체 엔진을 통해 효율적인 질의 실행 계획 수립할 수 있도록 하는 것이 목적이다.

Impala는 하둡 환경에서 병렬 DBMS 기술을 위해 설계된 오픈 소스 기반의 SQL 엔진으로 맵리듀스를 이용하지 않고 자체 질의 엔진인 Impalad를 이용하여 SQL 파싱, 의미 분석, 실행 계획 수립 및 최적화를 수행한다. 또한 디스크가 아닌 메모리에서 SQL을 처리하는 엔진으로 질의 처리 속도 향상시켰으며, 데이터 조회를 위한 인터페이스를 제공하는 것이 장점이다.

Tajo는 RDBMS에서 사용하는 SQL로 질의할 수 있는 하둡 기반 데이터웨어하우스 시스템으로 HDFS의 대용량 데이터 분석을 위해 맵리듀스 대신 자체 SQL 처리엔진을 내장시켰다. ETL뿐 아니라 Low-Latency, Long-Term 질의까지 지원하는 것이 장점이다.

Impala와 Tajo와 같은 자체 엔진을 사용하는 시스템은 SQL을 맵리듀스로 변환하는 방식이 아니라 자체적인 질의 실행 계획을 만들어 분산 노드에서 실행하기 때문에 맵리듀스 기반의 SQL on Hadoop 프레임워크보다 실행속도가 빠르다. 그러나 고도 분석 지원을 위한 최적화가 고려되기 어렵다.

2.3 SQL on Spark

SQL on Spark는 추상화된 분산 데이터 집합인 DataFrame에 관계 연산 인터페이스를 지원하는 기술과 SQL 질의처리를 최적화하는 기술, SQL 질의 결과(RDD)를 기계학습, 그래프 데이터 처리 등 고도 분석에 연계하는 기술로 구성된다.

Spark는 하둡의 맵리듀스 작업에서 성능의 병목현상으로 지목되던 디스크 I/O 비용을 최소화하고 데이터 분석 작업에 용이한 인메모리 컴퓨팅 기반의 범용적 데이터 분산처리 시스템이다. 또한, RDD라는 데이터 집합의 추상화 객체 개념을 도입하여 대용량 데이터에 대한 다양한 연산 작업이 가능하도록 한 것이 특징이다. RDD는 실제 물리적인 디스크에 저장된 데이터가 아닌 데이터 집합을 추상화하여 하나의 객체로 표현한다. 또한 여러 노드에 분산되어 다수의 파티션으로 관리되며 하나의 노드에 장애가 발생해도 데이터의 사용 이력 정보를 갖고 있는 lineage를 관리하여 복구 가능한 결함 포용성을 갖고 있다.

Spark는 RDD를 분산 클러스터의 메모리에 저장하기 때문에 기계학습, 그래프 처리 등 반복계산이 많은 작업에 대해 하둡보다 실행 속도가 빠르다. Spark에서 SQL을 지원하는 기술은 Hive에 의존적인 Shark로부터 시작되었으며, 현재는 Spark SQL에서 DataFrame과 SQL 질의처리 최적화 기인 Catalyst를 통해 지원한다.

Shark는 Spark 위에서 Hive의 HiveQL이 동작하도록 만든 인메모리 데이터 분석 시스템으로 Hive와 동일하게 맵리듀스를 질의처리 엔진으로 사용한다. Hive 사용자가 기존의 SQL을 수정하지 않고 그대로 사용 가능하지만, Hive가 가진 맵리듀스 한계를 동일하게 가지며, 계속 변경되고 새로운 기능이 추가됨에 따라 Shark도 변경해야하는 단점이 있다.

Spark SQL는 SQL, HiveQL, Scala로 표현된 관계형 질의가 Spark에서 실행되도록 DataFrame 인터페이스를 통해 평범한 RDD처럼 연산을 수행하거나 관계형 테이블에 적용할 수 있는 관계 연산을 지원한다. DataFrame를 테이블처럼 등록하면 추가적인 변환 없이 데이터에 SQL 질의를 실행할 수 있다.

Spark SQL은 고도 분석을 위한 기계학습을 위한 MLlib, 그래프 처리를 위한 GraphX 등과 같은 연계 모듈과 호환된다. 특히 기계학습 분야에서는 대용량 데이터 분석에 하둡의 머하웃(mahout)이 각광받고 있으나, MLlib은 Spark 환경에서 다양한 분석 방법을 지원하며 스트리밍이나 SQL과 연동할 수 있다. Spark는 데이터를 RDD 형태로 메모리에 로드하기 때문에 디스크를 사용하는 맵리듀스나 머하웃보다 10~100배까지 빠른 결과를 낼 수 있다.

그러나 SQL on Spark는 Impala와 Tajo와 마찬가지로 자체적인 질의처리 엔진을 갖고 있고 분석 대상 데이터를 인메모리에서 처리하기 때문에 기존 SQL on Hadoop 시스템보다 실행속도가 빠르나 중앙 집중식 처리에 의존하며 고도 분석 지원을 위한 최적화 기술이 고려되지 않는다.

3. 빅데이터 최적처리 프레임워크 설계

고도 분석 관점에서 기존의 Spark SQL 기술의 문제점은 크게 다음과 같이 네 가지로 요약할 수 있다.

- 2차 인덱스(secondary index)의 부재로 인해 DataFrame의 조건 연산을 포함한 검색 시 전체 데이터에 대한 스캔이 발생한다.
- Spark SQL의 핵심 기술인 Catalyst Optimizer는 고도 분석을 위해 규칙에 대한 확장성은 보장하나 기존 중앙집중식 DBMS에서 사용하는 최적화 모델을 그대로 적용한다.
- Spark SQL과 MLlib의 연동을 위해서는 DataFrame에서 RDD로의 타입 변환이 필요하며, 이 과정에서 고비용 변환처리가 발생한다.
- Spark SQL과 연동되는 MLlib에는 탐색 공간을 줄일 수 있는 공간 축소 기법(pruning)을 적용시킬 수 있는 기능이 없어 반복 작업이 비효율적이다.

따라서 본 논문에서는 이러한 문제점을 해결하기 위해 Fig. 2와 같이 분산/병행 처리 엔진 Spark SQL++와 반복적 작업 최적화 프레임워크를 설계한다.

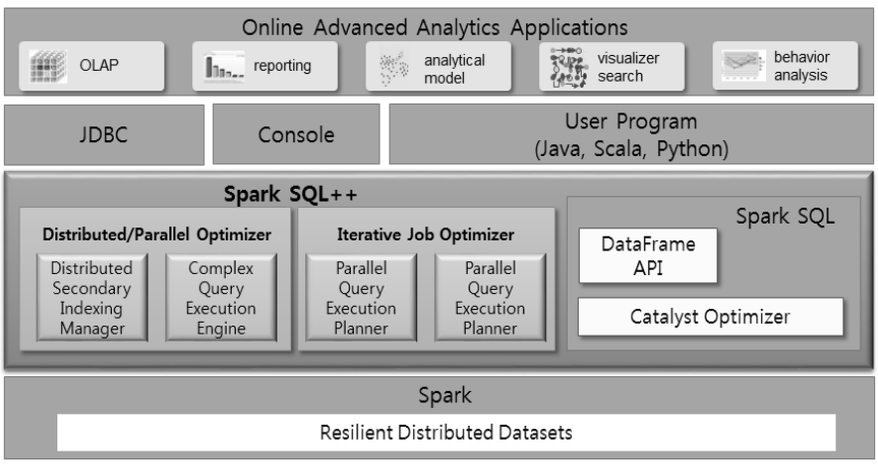

Fig. 2. The Architecture of Spark SQL++

3.1 분산/병행 처리를 고려한 SQL 질의처리 최적화 엔진

Spark SQL의 분산 2차 인덱스의 부재와 규칙 기반 최적와 비용 기반 최적화(조인 연산에만 적용)만 수행하는 문제점에 따라 고도 분석적 질의처리 최적화 엔진은 조건 연산의 최적화에 고려와 Complex SQL 질의 처리에 대한 고려가 필요하다.

1) 조건 연산의 최적화를 위한 분산 2차 인덱싱

Spark SQL++는 데이터 타입을 파악하여 DataFrame의 균등하게 데이터를 분할하여 파티션을 구성하고 각 파티션별 역(inverted) 인덱싱 및 계층적 분산 인덱싱을 수행한다. 이를 위해서 우선 고도 분석에서 필수적으로 요구되는 다양한 타입이 포함된 데이터 소스에서 DataFrame의 파티션 단위로 균등 분할할 방법이 필요하다. 예를 들어, 정수나 문자열에 대한 척도는 자체의 Point 값으로 단순 유클리드 거리를 계산하고 시간과 같이 범위를 갖는 데이터 타입은 Line에 대한 범위를 고려한 척도를 사용할 필요가 있다. 따라서 데이터 타입에 대해 값의 특성을 분석하고 유사성을 계산할 수 있는 거리 척도(distance metric)를 사용한다.

DataFrame의 특성을 고려하여 각 키에 대해 균등하게 분산시키기 위해서는 그리디(greedy) 알고리즘 기반 데이터 분할 알고리즘을 사용한다. 특히 분할 알고리즘은 균일(uniform)보다 비균일적(skewed) 데이터 분포에서 일부 키(key)에 편중되는 문제점을 해결하도록 설계하여 Spark의 워커 노드들에 균등하게 분산되도록 관리한다.

Spark SQL에서 범위 조건 검색을 지원하기 위해서는 데이터 타입을 기반으로 각 속성 별 범위에 대해 순차 접근 비용을 최소화하기 위해 파티션 기반 역(inverted) 인덱싱 및 계층적 분산 인덱싱을 수행한다. 역 인덱싱은 Spark SQL의 조건 연산을 사용하는 검색 질의처리 시 전체 스캔에 대한 비용을 최소화하기 위해 반드시 필요하다.

범위 속성을 가지는 데이터 타입에서 범위 조건 검색에 대해 접근성 향상을 지원하는 역 인덱싱과 계층적 분산 인덱싱 기법은 핵심 속성을 로우 키로 갖는 컬럼 패밀리 기반의 역 데이터 인덱스를 구조화 한다. 그리고 검색 공간(search space) 축소를 위한 계층적 분산 인덱스 구조를 정의한다. DataFrame의 파티션들을 하나의 인덱스 노드로 매핑하여 계층적 트리 형태로 데이터를 구조화한다. 그리고 유사한 키를 가진 균등한 파티션들의 그룹을 노드로 추상화하고 계층화한다. Spark의 워커 노드들이 인덱스 노드를 관리하고 높이균형트리 구조를 유지하기 때문에 트리 탐색의 Worst-case 비용을 로그 시간으로 감소시킬 수 있다.

2) Complex SQL 질의처리를 위한 분산/병행 최적화

Complex SQL 질의는 조건을 수반한 SELECT 연산과 같은 단순 질의가 아닌 파티션에 포함된 전체 데이터에 대한 다수의 조인과 집계, 정렬이 필요한 SQL 질의이다. 이를 위해 제안하는 프레임워크에서는 Fig. 3과 같이 최적화된 실행 계획 생성을 위한 Complex SQL 질의의 분산/병행 트리를 생성하고 최적화된 물리적 계획 선택을 위한 분산/병행 비용 모델에 기반한 질의실행 계획을 생성한다.

우선 최적화된 실행 계획 생성을 위한 Complex SQL 질의의 분산/병행 트리를 생성하기 위해 분산/병행 기반 2단계 로컬 및 글로벌 실행 계획 트리 구조를 작성한다.

기존의 Catalyst 최적화기는 기존 DBMS에서 사용하는 단인 트리 구조로의 SQL 질의를 논리적/물리적 계획으로 변환하고 Lazy 실행을 통해 최적화를 진행하나, 기존의 방식은 여러 워커 노드들로 구성되는 클러스터 환경에서 정밀한 실행 계획을 생성하는데 한계가 있다. 반면 제안하는 최족화 프레임워크는 정밀한 최적화를 진행하기 위해 로컬 실행 계획 트리와 글로벌 실행 계획 트리로 구성된 2계층의 분산/병행 트리 구조를 생성한다. 또한 로컬과 글로벌 트리 노드들 간의 연결성을 파악하고 노드들의 타입을 규정한다.

분산/병행 트리 구조를 바탕으로 SQL 질의를 로컬 트리와 글로벌 트리로 변환된다. 먼저 SQL 질의에 대한 분산/병행 기반 분석을 진행하고 이것을 논리적 계획으로 변환하는 규칙과 분산/병행 트리 형태로 생성된 논리적 계획을 통해 다수의 물리적 계획을 생성하는 변환 규칙이 적용된다.

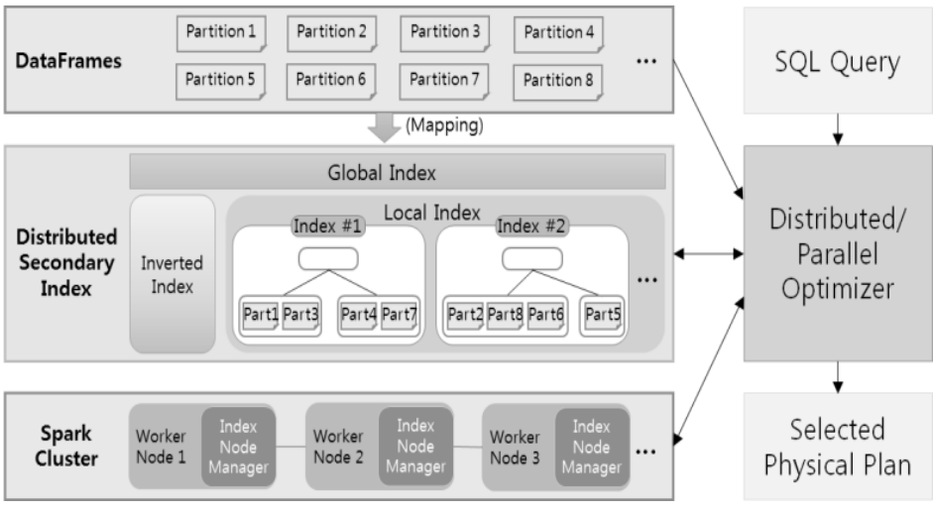

Fig. 3. SQL Query Processing Framework for Complex SQL

논리적/물리적 계획의 분산/병행 트리 구조화 이후 이를 기반으로 최적화된 물리적 계획 선택을 위한 과정이 수행된다. 따라서 물리적 계획에 대한 각 단계(layer)별 실행 계획 트리에 대한 비용 모델을 사용한다. 비용 모델은 물리적 계획에 대한 각 단계(layer)별 실행 계획 트리에 대한 비용 모델과 로컬과 글로벌 통합 실행 계획 트리에 대한 비용 모델로 구분된다.

물리적 계획에 대한 각 단계(layer)별 실행 계획 트리에 대한 비용 모델은 Complex SQL 질의 변환 결과인 로컬과 글로벌 각각 실행 계획 트리에 대해 질의처리 비용을 계산하는 모델이다. 로컬 실행 계획 트리의 비용은 하나의 워커 노드에서 발생하는 비용을 의미하며 글로벌 실행 계획 트리 비용은 다수의 워커 노드들에 속한 DataFrame의 파티션들에 대한 연산처리 비용을 나타낸다. 글로벌 실행 계획 트리의 경우 분산 인덱스를 통해 절감된 비용을 반영하여 모델을 개발하고 로컬 실행 계획 트리의 경우 해당 워커 노드의 자원 상황을 고려한 모델이다.

로컬과 글로벌 통합 실행 계획 트리에 대한 비용 모델은 정확한 비용 모델 구축하기 위해 각각의 로컬과 글로벌 비용 모델을 바탕으로 최적화된 물리적 계획을 최종 선택하기 위한 통합 비용 모델이다. 우선 단순 로컬과 글로벌의 비용 합산으로 계산하는 기본 방법을 먼저 구축하고 통계적 결과 데이터를 바탕으로 기본 방법보다 최적화가 가능한 비용 모델을 도출한다. 또한, 질의처리 시 병목 현상이 발생하는 지점을 찾고 그에 따른 통합 실행 계획 비용 모델에 반영하여 보다 저비용 처리가 가능한 최적화된 물리적 계획을 선택한다.

3.2 관계형 연산을 지원하는 기계학습 최적화

기존의 Spark SQL과 MLlib의 연동을 위해서는 DataFrame에서 RDD로의 타입 변환이 동반되며 이 과정에서 타입 변환 비용이 발생한다. 또한 MLlib에는 탐색 공간을 줄일 수 있는 공간 축소 기법을 적용시킬 수 있는 기능이 없어 반복 작업에 병목 현상이 발생할 경우 데이터 처리 비용이 높아진다.

따라서 최적화 프레임워크에서 관계형 연산을 지원하는 기계학습 최적화 시스템은 크게 두 가지 기능을 수행한다. 첫째, 관계형 연산을 지원하는 기계학습 통합 엔진은 통합 기계학습 엔진을 위한 병렬 관계형 연산을 지원한다. 둘째, 분산 환경에서 관계형 연산 기반의 탐색 공간 병렬 축소기능을 지원한다.

통합 기계학습 엔진은 기계학습 엔진 개발과 분산된 노드에서 통합된 관계형 연산을 지원한다. 최적화 프레임워크의 기계학습 엔진은 분산된 노드에서 관계형 연산을 병렬적으로 처리하고 기계학습 모듈에 관계형 연산 활용을 할 수 있도록 데이터를 연동한다. 기계학습 모듈에서 관계형 연산을 이용하면 데이터 선택, 제거, 샘플링 등의 기능을 직관적이고 신속하게 수행 가능하다.

MLlib의 기계학습 알고리즘 클래스에 일단 DataFrame 타입의 데이터가 전달되면, 관계 연산을 통해서 필요한 데이터를 반환받을 수 있다. MLlib는 RDD 연산을 통해 데이터에 접근하도록 작성되어 있다. 따라서 DataFrame 타입으로 데이터를 로드한 경우에도 그대로 동작되도록 해야 하기 때문에 MLlib의 데이터 로드 클래스가 DataFrame 타입으로 데이터 로드 시, 기존 MLlib 코드의 변경을 최소화하는 RDD 연산을 재정의 한다.

관계형 연산 기반의 탐색 공간 축소 기법은 기계학습 알고리즘의 반복 작업에서 전체 데이터의 탐색 공간을 최소화함으로써 성능을 향상시킨다. 그러나 Spark와 같은 분산 환경에서 탐색 공간을 병렬적으로 푸르닝하기 위해서는 DataFrame의 관계 연산을 이용한 푸르닝과 MLlib 파이프라인을 통합하기 위한 고려가 필요하다.

최적화 프레임워크는 DataFrame의 관계 연산을 이용하여 DataFrame의 데이터를 부분 검색하여 데이터 탐색 공간을 축소한다. 한 번의 반복 작업이 종료되면서 검색 조건은 변경될 수 있으므로 고정된 관계 연산을 설정하는 것이 아니

라, 동적으로 설정하게 된다. 공간 축소 조건은 데이터가 가지고 있는 값이나 앞 단계에의 반복 작업이 실행되면서 계산된 값을 기준으로 재계산될 수 있으므로 이러한 값들을 조합하거나 재계산하여 DataFrame의 관계 연산을 재구성한다.

DataFrame에 공간 축소 기법을 적용한 후, 기계학습 파이프라인 통합을 수행한다. MLlib에서는 기계학습 알고리즘의 데이터 처리 단계를 파이프라인으로 처리할 수 있게 제공하고 있으며, 기계학습 파이프라인에 관계 연산을 통해 데이터를 선택할 수 있는 DataFrame를 추가하여 공간 축소 기법이 추가된 파이프라인이 이용된다. 이를 위해 최적화 프레임워크는 Spark에서 제공하는 파이프라인 API를 확장하여 DataFrame의 관계 연산을 포함하는 공간 축소 기법이 포함된 파이프라인을 사용한다.

4. 결 론

빅데이터의 고도 분석 지원 기술은 공공, 민간, 기업에서 매우 중요한 가치를 가지며 앞으로 SQL 기반 플랫폼에서 필수적으로 요구되고 있어 빅데이터 처리에서 Spark 플랫폼은 기존 하둡 플랫폼을 이미 넘어 그 중요성과 가치가 매우 증대되고 있다.

따라서 본 논문에서는 설계한 Spark SQL++와 최적화 프레임워크는 기존 Spark SQL의 고도 분석 작업에서의 한계와 문제점 극복하고 2차 인덱싱 기법과 복합 질의 처리를 최적화를 수행한다. 또한 요청된 질의의 처리 비용을 예측할 수 있는 비용 모델을 제시하고 이를 바탕으로 작업 최적화 기법과 분산/병행 처리 관리자를 제공한다.

빅데이터의 고도 분석에 특화된 질의 처리 엔진과 최적화 프레임워크 개발을 통해 빅데이터 분석의 기틀을 제공하여 다양한 연구 분야(사회 구조 분석, 지리 분석, 바이오 인포매틱스 등)에 활용 가능해 질 것으로 예상된다.

References

[1] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” *Sixth Symposium on Operating System Design and Implementation*, 2004.

[2] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins “Pig latin: a not-so-foreign language for data processing,” *SIGMOD*, pp.1099-1110, 2008.

[3] A. Thusoo, J. S. Sarma, N. Jain, and R. Murthy et al., “Hive - A Petabyte Scale Data Warehouse Using Hadoop,” *International Conference on Data Engineering*, 2010.

[4] H. Choi et al., “Tajo: A distributed data warehouse system on large clusters,” *International Conference on Data Engineering*, 2013.

[5] M. Kornacker et al., “Impala: A Modern, Open-Source SQL Engine for Hadoop,” *CIDR*, 2015.

[6] M. Zaharia et al., “Spark: Cluster Computing with Working Sets,” *HotCloud*, 2010.

[7] M. Armbrust et al., “Spark SQL: Relational Data Processing in Spark,” *SIGMOD*, pp.1383-1394, 2015.

정 재 화

e-mail : jaehwachung@knou.ac.kr

1999년 고려대학교 컴퓨터교육과(이학사)

2011년 고려대학교 컴퓨터교육과 (이학석・박사)

2012년～현 재 한국방송통신대학교 컴퓨터과학과 조교수

관심분야: 공간질의처리 및 인덱싱, 분산 컴퓨팅플랫폼 (mapreduce, spark), 모바일 데이터 관리, RFID, 무선 센서 네트워크