SQL-맵리듀스 통합을 위한 질의 인터페이스 및 질의 최적화 연구 현황

이태휘*, 정문영*, 임동혁**, 원종호*
*한국전자통신연구원
**호서대학교 컴퓨터공학과
e-mail : taewhi@etri.re.kr, mchung@etri.re.kr, dhim@hoseo.edu, jhwon@etri.re.kr

A Survey on Query Interface and Query Optimization for the SQL-MapReduce Integration

Taewhi Lee*, Moonyoung Chung*, Dong-Hyuk Im**, Jongho Won*
*Electronics and Telecommunications Research Institute
**Dept of Computer Engineering, Hoseo University

요 약

SQL의 사용자 편의성과 맵리듀스의 함수형 분산 처리 능력을 동시에 활용하기 위한 통합 작업이 이루어지고 있다. 본 논문에서는 최근 이루어진 관련 연구를 질의 인터페이스 측면과 질의 최적화 측면으로 나누어 살펴보고, 향후 연구 과제에 대해 알아본다.

1. 서론

맵리듀스(MapReduce)[1]는 분산 처리 시 공통으로 필요한 일들, 예를 들면, 실행 프로그램의 배포, 분산 실행 작업의 스케줄링, 수행에 실패한 작업 처리 등의 기능을 지원하여 대용량 데이터 처리에 대한 진입 장벽을 획기적으로 낮추었으며, 맵리듀스를 구현한 오픈 소스인 하둡(Hadoop)[2]의 등장으로 인해 급속히 확산되었다. 사용자는 이전보다 훨씬 쉬워진 프로그래밍 API를 통해 분산 처리를 하고자 하는 동작을 맵(Map) 함수와 리듀스(Reduce) 함수로 구현하면 된다.

하지만 결국에는 프로그래밍 과정이 필요하기 때문에 임의(ad-hoc)의 작업을 실행하는 데에는 불편하며, 프로그래밍에 익숙하지 않은 데이터 분석가가 사용하기에는 여전히 어려운 측면이 있다. 또한 구조화된 형태의 작업은 오랜 기간 동안 연구되고 활용되어 온 SQL로 표현하는 것이 더 효율적이다.

이러한 문제를 개선하기 위해, 사용자에게 쉽고 익숙한 선언형(declarative) 언어인 SQL과 조금 더 번거롭기는 하나 분산 처리에 적합하고 더 자유롭게 함수 형태로 표현 가능한 맵리듀스를 통합하는 방향으로 질의 인터페이스 확장 및 통합 질의 최적화 연구가 진행되고 있다. 본 논문에서는 이러한 SQL-맵리듀스 통합 질의 관련 연구를 살펴보고 향후 연구 방향에 대해 생각해 본다.

2. 연구 현황 및 향후 연구

2.1. SQL-맵리듀스 통합 질의 인터페이스

SQL과 맵리듀스를 통합하려는 노력은 테라데이터(Teradata), 애스터데이터(Aster Data Systems), 오라클(Oracle) 등 상용 DBMS를 보유한 기업들에 의해 주로 이루어졌다. 기존 DBMS에 분산 처리 능력을 강화하기 위한 노력의 일환이었다.

테라데이터는 병렬 DBMS 제품인 테라데이터 EDW(Enterprise Data Warehouse)의 데이터와 하둡의 데이터를 서로 교차하여 사용 가능한 인터페이스를 제공한다[3]. SQL 질의에서는 테이블 UDF(User-Defined Function)를 통해 하둡 데이터에 접근 가능하며, 반대로 하둡 맵리듀스에서는 DBInputFormat 및 TeradataInputFormat 등의 입력 관련 클래스를 활용하여 테라데이터 EDW의 데이터에 접근 가능하다. 이 방법은 각기 다른 시스템에 저장되어 있는 데이터를 함께 접근하여 처리하게 해주나, 데이터 처리는 SQL 또는 맵리듀스 중 하나로 수행하도록 분리되어 있다.

애스터 데이터는 맵리듀스를 UDF로 활용하는 SQL/MapReduce[4]를 개발하였다. SQL/MapReduce에서 사용자는 제공되는 고유한 자체 자바 API를 통해 맵 함수와 리듀스 함수에 대응되는 로우 함수(row function)와 파티션 함수(partition function)를 구현한다. 구현한 함수는 FROM 절에서 사용 가능하며, 함수의 결과를 일반 테이블처럼 취급하여 중첩 사용도 가능하다. SQL과 맵리듀스를 혼합하여 전체 질의를 수행할 수 있다는 장점이 있으나, 하둡의 소스 코드와는 호환되지 않는 자체 API로 UDF를 구현해야 하는 단점이 있다. 오라클 인-데이터베이스 하둡(Oracle In-Database Hadoop)[5]에서는 하둡 자바 소스 코드로 된 맵/리듀스 함수를 거의 수정 없이 실행 가능하도록 하여 이러한 호환성 문제를 개선하였다. 맵 함수와 리듀스 함수는 파이프라인 테이블 함수로 구현되는데, 파이프라인 테이블 함수는 데이터 흐름에 삽입되

어 중간 데이터를 저장하지 않고 데이터를 SQL 문장으로 스트리밍하게 해준다. 소스 코드는 하둡과 호환이 되지만, 함수의 실행은 하둡을 사용하지 않고 오라클의 JVM에서 이루어지며, 추가된 TableReader와 TableWriter 클래스를 통해 SQL의 데이터 타입과 Hadoop의 Writable 데이터 타입 간의 변환을 처리한다.

한편, 하이브(Hive)[6]는 SQL과 흡사한 HiveQL 질의를 자체적으로 맵리듀스 프로그램으로 변환하여 실행하는 프레임워크로, 질의에 별도의 맵리듀스 스크립트를 질의에 삽입할 수 있는 로우 기반 스트리밍 인터페이스를 지원한다. 삽입되는 스크립트는 로우를 표준 입출력으로 읽고 쓰기만 하면 작성 언어에 무관하게 사용 가능하다. 하이브는 앞에서 살펴본 다른 DBMS나 최근 개발된 임팔라(Impala)[7], 타조(Tajo)[8] 등 여러 SQL-온-하둡(SQL-on-Hadoop) 시스템과 달리 자체 실행 엔진이 없고 하둡 맵리듀스(혹은 상응하는 다른 실행 엔진) 상에서 동작하는 형태이기 때문에, 통합보다는 변환 실행에 가깝다고 볼 수 있다.

이와 같이 현재까지 SQL과 맵리듀스의 인터페이스를 통합하려는 시도들은 한 시스템 내에서 SQL과 맵리듀스를 함께 실행하는 방향으로 진행되어 왔다. 그러나 얀(YARN)[9], 메소스(Mesos)[10] 등 클러스터 통합 자원 관리 프레임워크가 등장함에 따라, 최근에는 이를 일종의 "데이터 운영 체제"[11]로 간주하고 사용 자원을 공유하며 데이터 처리 특성에 맞게 여러 애플리케이션 프레임워크를 활용하는 추세로 나아가고 있다. 이러한 흐름에 발맞추어, 다중 시스템을 활용함으로써 소스 코드 단계를 넘어 맵리듀스, 더 나아가 다른 여러 프레임워크의 프로그램 바이너리를 그대로 실행하는 형태의 통합에 관한 연구가 필요하리라 생각한다.

2.2. SQL-맵리듀스 통합 질의 최적화

2.1절에서 살펴본 대로, SQL과 맵리듀스를 통합한 질의의 형태는 질의는 SQL에 맵리듀스를 UDF로 활용하는 방식이 주를 이루고 있다. 본 절에서는 맵리듀스를 UDF로 간주하고 여러 개의 맵리듀스 UDF를 수행하는 질의의 실행 계획을 최적화하는 연구들에 대해 알아본다.

관련 연구는 크게 UDF를 블랙 박스(black-box)로 간주하는 방법과 UDF 내부를 분석하는 방법의 두 방향으로 나눌 수 있다. UDF를 블랙박스로 간주하는 방법은 데이터를 본격적으로 분산 처리하기 이전에 일부 처리 작업만 시범적으로 실행하여 선택도 등의 통계 정보를 수집한 후 이를 활용하여 비용 기반으로 다음 실행할 연산을 선택하는 방법이다[12]. 이를 실현하기 위해서는 동적으로 실행 계획에 따라 수행하는 기능이 뒷받침되어야 한다. 이 방법은 초기 오버헤드가 발생하는 단점이 있으나, 일반적으로 적용이 가능하고 실행 계획의 품질이 일정 수준 보장된다. 이와 달리, 맵리듀스 UDF의 소스 코드를 정적 분석을 통해 파악하여 최적화를 수행하는 방법도 있다[13,14]. 일반적인 프로그램을 분석하여 실행 비용을 예측한다면 이상적이나 매우 어려운 일이며, 현재로서는 제한적인 API를 사용하는 프로그램에 한해서 분석 및 최적화가 가능한 수준에서 연구가 진행되고 있다. 여러 맵/리듀스 함수 중 함수 내에서 발생하는 입출력 형태를 파악하여 실행 순서를 변경해도 결과가 동일한 함수들을 찾고, 이러한 경우들을 모두 고려하여 실행 순서를 최적화한다.

UDF는 동작이 한정되어 있지 않기 때문에 그만큼 동작의 결과를 예측하기가 어려움은 자명하다. 질의 최적화의 개선과 더불어, 질의에 부가적인 힌트 정보를 제공하거나 이전에 실행한 질의의 수행 정보를 향후 질의의 최적화에 활용하는 등 다른 각도에서의 접근이 필요하다고 생각한다.

3. 결론

본 논문에서는 SQL과 맵리듀스의 통합을 위한 질의 인터페이스와 질의 최적화 연구 현황을 간략히 소개하였다. 데이터 특성과 주요 질의 형태에 따라 다양한 빅데이터 처리 프레임워크가 등장, 발전하고 있는 만큼, 향후에는 이러한 연구를 바탕으로 하나의 시스템에 국한되지 않고 이를 조합해서 활용할 수 있는 형태의 일반적이고 유연한 통합 인터페이스 및 최적화 연구가 필요할 것이다.

Acknowledgment

이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구(No. NRF-2014R1A1A1002236)이며, ETRI R&D 프로그램 ("듀얼모드 배치-쿼리 분석을 제공하는 빅데이터 플랫폼 핵심 기술 개발, 15ZS1400")의 일환으로 수행됨.

참고문헌

[1] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified data processing on large clusters," Proc. of OSDI, 2004.

[2] http://hadoop.apache.org

[3] Yu Xu et al., "Integrating Hadoop and parallel DBMS," Proc. of SIGMOD, 2010.

[4] Eric Friedman et al., "SQL/MapReduce: A practical approach to self-describing, polymorphic, and parallelizable user-defined functions," Proc. of VLDB Endowment, 2009.

[5] Xueyuan Su and Garret Swart, "Oracle In-Database Hadoop: When MapReduce meets RDBMS," Proc. of SIGMOD, 2012.

[6] Ashish Thusoo et al., "Hive - A petabyte scale data warehouse using Hadoop," Proc. of ICDE, 2010.

[7] Marcel Kornacker et al., "Impala: A modern,

open-source SQL engine for Hadoop," Proc. of CIDR, 2015.

[8] Hyunsik Choi et al., "Tajo: A distributed data warehouse system on large clusters," Proc. of ICDE, 2013.

[9] Vinod Kumar Vavilapalli et al., "Apache Hadoop YARN: Yet another resource negotiator," Proc. of SoCC, 2013.

[10] Benjamin Hindman et al., "Mesos: A platform for fine-grained resource sharing in the data center," Proc. of NSDI, 2011.

[11] http://www.hortonworks.com/labs/yarn

[12] Konstantinos Karanasos et al., "Dynamically optimizing queries over large scale data platforms," Proc. of SIGMOD, 2014.

[13] Fabian Hueske et al., "Opening the black boxes in data flow optimization," Proc. of VLDB Endowment, 2012.

[14] Fabian Hueske et al., "Peeking into the optimization of data flow programs with MapReduce-style UDFs," Proc. of ICDE, 2013.