

Spark SQL에서 셔플 파티션 수에 영향을 받는 쿼리 특성 분석*

이유림⁰¹ 윤현식² 정연돈²

¹ 고려대학교 사이버국방학과

² 고려대학교 컴퓨터학과

leeyou6757@korea.ac.kr, hyunsikyoon@korea.ac.kr, ydchung@korea.ac.kr

Analysis of Query Characteristics Affected by Shuffle Partitions in Spark SQL

Yu Rim Lee⁰¹ Hyun Sik Yoon² Yon Dohn Chung²

¹Department of Cyber Defense, Korea University

²Department of Computer Science and Engineering, Korea University

요 약

Spark는 현대에서 빅데이터를 처리하는 프레임워크로써 널리 사용되고 있다. Spark에서 머신러닝을 활용하여 configuration 튜닝을 하기 위해서는 샘플 데이터 양을 줄여야 하고, 이를 위해 중요하지 않은 쿼리를 제거하는 방법은 유용하게 사용된다. 이 논문에서는 주요 configuration인 셔플 파티션 수에 영향을 받는 쿼리들의 특징을 분석했다. ‘order by’ 구문을 주로 사용하는 쿼리가 영향을 많이 받는 것으로 나타났다. 이러한 쿼리들은 쿼리 제거 단계에서 우선적으로 제외하여 시간을 단축시킬 수 있다.

1. 서 론

Hive[1], Spark SQL[2]와 같은 빅데이터 쿼리 시스템과 Hadoop, Spark[3]와 같은 프레임워크는 현대에서 빅데이터를 저장하고 처리 및 분석하는 데 사용된다.

이 중, Spark SQL은 빠른 속도라는 장점으로 널리 쓰이고 있지만 최적의 성능을 위해서는 configuration을 튜닝하여 최적화한 후 사용해야 한다. configuration은 Spark의 동작을 제어하기 위한 변수로, 메모리 크기를 지정하는 spark.executor.memory, CPU 코어 수를 결정하는 spark.executor.cores를 비롯한 300개 이상의 configuration이 존재한다. configuration 튜닝에 머신러닝을 사용해 이를 해결하려는 연구가 계속되고 있지만, 머신러닝 기반의 자동 튜닝은 매우 많은 양의 학습 데이터를 필요로 한다. 대용량 데이터를 다루는 Spark의 특성상, 실제 어플리케이션을 사용하여 데이터를 수집해야 하기 때문에 시간이 많이 소요된다.

이 문제점을 해결하기 위한 LOCAT[4]에서는 QCSA(Query Configuration Sensitivity Analysis) 과정을 통해 configuration에 영향을 상대적으로 적게 받는 쿼리들을 제거한다. 쿼리를 제거하면 테스트해야 하는 쿼리의 개수가 줄어들기 때문에 데이터 수집 시간을 단축시킬 수 있다. QCSA는 LOCAT[4]에서뿐만 아니라 샘플로 사용할 데이터를 줄이는 데 있어 유용하기 때문에 여러 시스템에서 널리 사용된다.

그러나 QCSA 과정 역시 데이터 수집을 위해서는 어플리케이션을 여러 번 반복해서 실행해야 한다. Spark configuration 중 sql.shuffle.partitions는 데이터를 섞는 셔플링 작업을 수행할 때 생성되는 셔플 파티션의 개수를 결정한다. 본 논문에서는 sql.shuffle.partitions configuration을 ‘셔플 파티션 수’라고 한다. 이때 셔플링에 사용되는 파티션 수가 너무 적으면 파티션의 크기가 커져서 작업이 느려지거나 OutOfMemoryError 등의 오류가 발생할 수 있으며, 반대로 파티션 수가 너무 많으면 네트워크 대역폭을 낭비하고 작업 시간이 길어질 수 있다[5]. LOCAT[4]에서는 데이터 크기에 관계없이 모든 configuration 중 셔플 파티션 수가 가장 중요한 것으로 나타났다. 따라서 이 논문에서는 주요 configuration인 셔플 파티션 수에 영향을 받는 쿼리들의 특징을 각각

* 이 논문은 1. 2023년도 과학기술정보통신부의 재원으로 정보통신기획평가원-정보통신·방송 기술개발사업 (No.2021-0-00634), 2. 2023년도 정부(교육부)의 재원으로 한국연구재단 (No. NRF-2021R1A6A1A13044830)의 지원을 받아 수행된 연구임

분석한다. 이 분석 결과를 통해 쿼리의 특징만으로 각 쿼리의 중요도를 예측하여 QCSA의 시간을 단축시킬 수 있다.

2. 시스템 환경 및 데이터

Spark 3.3.2 version을 사용하여 진행했다. 쿼리의 수행시간은 실행되는 환경에 영향을 많이 받기 때문에 실제로 주로 사용되는 Hadoop-Yarn Cluster Mode를 사용했다. 1개의 Master node와 3개의 Worker node를 생성했고, 각각에는 8192MB 메모리를 할당했다.

데이터셋과 워크로드로는 TPC-DS 벤치마크를 사용했다. TPC-DS 벤치마크는 데이터베이스 시스템의 성능을 평가하는데 사용되는 벤치마크이다. 본 연구에서는 10GB 규모의 데이터셋을 사용하여 총 100개의 쿼리에 대한 Spark SQL의 성능을 평가했다.

3. 분석 방법

서플 파티션 수에 수행시간이 영향을 많이 받는 쿼리들의 특징을 파악하고자 했다. 총 50개의 무작위 서플 파티션 수 값에 대한 100개의 쿼리의 실행시간을 각각 기록했다. 자세한 분석 순서는 다음과 같다.

1) spark application을 실행하는 동안의 각각의 query의 실행 시간들을 기록한다. 총 100개의 쿼리에 대해 50번 실행하였다.

j 번째 실행에서의 i 번째 쿼리 q_{ij} 를 돌린 실행시간은 다음과 같이 표현한다.

$$t_{ij}(i = 1, 2, \dots, 100; j = 1, 2, \dots, 50)$$

2) q_{ij} 의 coefficient of variation을 이용해 각 쿼리의 configuration 민감도(sensitivity)를 계산한다.

$$sensitivity(q_i) = \frac{\sigma_i}{\mu_i}$$

$$\mu_i = \frac{\sum_{j=1}^{50} t_{ij}}{50}, \sigma_i = \sqrt{\frac{\sum_{j=1}^{50} (t_{ij} - \mu)^2}{50}}$$

3) 민감도가 높은 쿼리들은 서플 파티션 수에 영향을 많이 받고, 민감도가 낮은 쿼리들은 영향을 거의 받지 않는다. 민감도가 큰 상위 10개의 쿼리를 sensitive query로 분류해 분석했다.

4. 분석 결과

쿼리의 민감도 분포는 다음 [그림 1]과 같다. x축은 쿼리 번호, y축은 각 쿼리에 해당되는 민감도이다. 대체로 민감도가 (0, 0.1) 사이에 쿼리들이 많이

분포하고 있음을 알 수 있으며, 민감도가 (0.3, 0.6)으로 높은 쿼리들이 소수 존재했다. 따라서 민감도가 높은 상위 10개의 쿼리를 추출했다. TPC-DS의 query20, query21, query32, query36, query37, query39, query94, query95, query96, query97이 해당되었다.

이 쿼리들의 특징은 다른 쿼리들에 비해 ‘order by’ 명령어를 많이 사용한다는 것이다. ‘order by’ 명령어를 사용하는 횟수의 분포는 그림 2와 같다. x축은 ‘order by’ 구문을 사용한 횟수, y축은 민감도이다. 삼각형은 민감도의 평균, 사각형 내부의 선은 중앙값을 의미한다. 민감하지 않은 쿼리들이 평균 1.43개의 ‘order by’ 명령어를 사용하였고, 민감도가 높은 쿼리는 평균 3.1개의 약 두배에 달하는 order by 명령어를 사용하였다. 민감도와 ‘order by’개수의 관계는 그림 2과 같다. ‘order by’ 구문의 사용 횟수가 늘어날수록 쿼리의 평균 민감도도 높아지는 것을 알 수 있다.

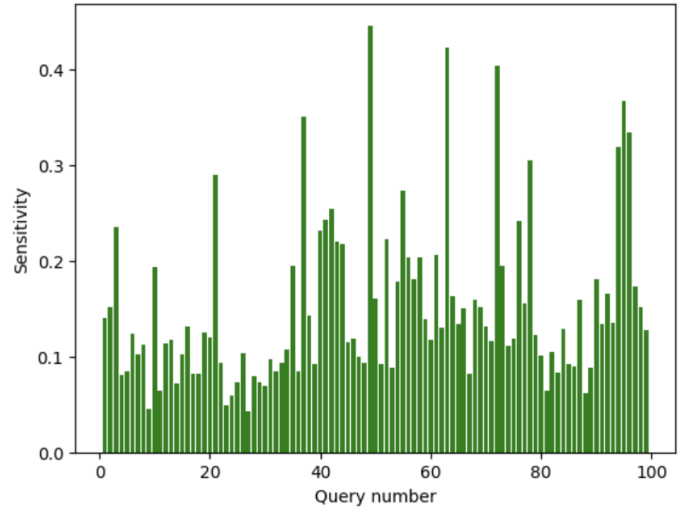


그림 1 TPC-DS 쿼리 민감도 분포

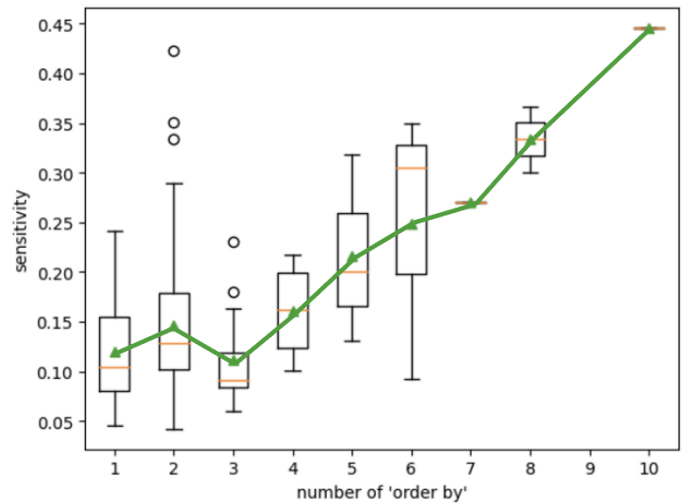


그림 2 order by 사용 횟수와 민감도의 관계

‘order by’ 구문은 셔플 파티션 수의 영향을 많이 받는 것으로 나타났다. order by 구문이 영향을 미치는 이유로는 다음과 같이 설명할 수 있다. ‘order by’ 명령어는 모든 데이터를 정렬하기 위해 데이터를 파티션 단위로 나눈 다음, 각 파티션을 정렬하고 병합하는 방식으로 처리된다. 이 과정에서 셔플 파티션 수는 정렬에 사용되는 파티션 수를 결정한다. 이 값이 너무 작거나 크면 order by 구문을 실행할 때 성능에 영향을 미칠 수 있다. 1장에서 언급했듯, 파티션 수가 너무 적으면 정렬에 사용되는 리소스가 부족해지므로 처리 속도가 느려질 수 있다. 반면 파티션 수가 너무 많으면 데이터 이동이 많아져 처리 속도가 느려질 수 있다.

5. 결론 및 향후 연구

본 논문에서는 셔플 파티션 수에 수행 시간이 큰 영향을 받는 쿼리들의 특징을 분석했다. 분석 결과 ‘order by’ 구문을 많이 사용하는 쿼리는 configuration tuning에 중요한 쿼리로 나타났고, QCSA 단계 시 미리 제외하여 시간을 단축시킬 수 있다.

본 논문에서는 하나의 parameter에 대한 분석만을 진행하였지만, 향후 다른 configuration parameter에 대한 분석도 진행한다면, 실제로 많은 시행을 반복하지 않고도 쿼리의 특징만을 이용해 QCSA를 진행할 수 있을 것이다.

참 고 문 헌

- [1] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. Hive: A Warehousing Solution over a Map-Reduce Framework, Proc. VLDB Endow. 2, 2 (aug 2009), 1626–1629 2009.
- [2] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. Spark SQL: Relational Data Processing in Spark. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (Melbourne, Victoria, Australia) (SIGMOD '15). Association for Computing Machinery, New York, NY, USA, 1383–1394. 2015.
- [3] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica, Spark: cluster computing with working sets, HotCloud'10: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing ,2010.
- [4] Jinhan Xin, Kai Hwang, Zhibin Yu, LOCAT: Low-Overhead Online Configuration Auto-Tuning of Spark SQL Applications, SIGMOD '22: Proceedings of the 2022 International Conference on Management of Data, 16 pages. 2022
- [5] Mayuresh Kunjir and Shivnath Babu. Black or White? How to Develop an AutoTuner for Memory-Based Analytics. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (Portland, OR, USA) (SIGMOD'20), 2020.