Spark 클러스터 환경에서 분산 처리 성능 향상을 위한 Buffer 최적화 시스템 연구

홍석민[1], 이소영[1], 신용태[2]
[1]숭실대학교 컴퓨터학과
[2]숭실대학교 컴퓨터학부
ghdtjrals3@gmail.com
2soyeong10@gmail.com
shin@ssu.ac.kr

A Study on Buffer Optimization System for Improving Performance in Spark Cluster

Seok-Min Hong[1],So-Yeoung Lee[1], [2]Yong-Tae Shin
[1]Dept. of Computer Engineering, SoongSil University
[2]School of Computing, SoongSil University

요 약

Statista 통계 조사에 따르면 데이터의 규모는 매년 증가할 것으로 예상하고 빅데이터 처리 프레임워크의 관심이 높아지고 있다. 빅데이터 처리 프레임워크 Spark는 Shuffle 과정에서 노드 간 데이터 전송이 일어난다. 이때 분산 처리한 데이터를 네트워크로 전송하기 위해 객체를 바이트 스트림으로 변환하여 메모리 buffer에 담는 직렬화 작업이 필요하다. 그러나 바이트 스트림을 buffer에 담는 과정에서 바이트 스트림의 크기가 메모리 buffer보다 클 경우, 메모리 할당 과정이 추가로 발생하여 전체적이 Spark의 성능 저하로 이어질 수 있다. 이에 본 논문에서는 Spark 환경에서 분산 처리 성능 향상을 위한 직렬화 buffer 최적화 시스템을 제안한다. 제안하는 방법은 Spark Driver가 Executor에게 작업을 할당하기 전 직렬화된 데이터 크기 측정과 직렬화 옵션 설정을 통해 Executor에게 적절한 buffer를 할당할 수 있다. 향후 제안하는 방법의 검증을 위해 실제 Spark 클러스터 환경에서 성능 평가가 필요하다.

1. 서론

Statista 통계 조사[1]에 따르면 데이터의 규모는 매년 증가할 것으로 예상하고 많은 데이터를 처리할 수 있는 빅데이터 처리 프레임워크의 관심이 높아지고 있다. 빅데이터 처리 프레임워크 Spark는 처리를 위해 분산된 데이터를 네트워크로 모은 다음, 데이터를 분할하는 작업인 Shuffle을 진행한다. 이때 분산 처리한 데이터를 네트워크로 전송하기 위해서는 객체를 바이트 스트림으로 변환하여 메모리 buffer에 담는 직렬화 작업이 필요하다.

그러나 직렬화 작업은 객체를 바이트 스트림으로 변환하여 buffer에 담는 과정에서 변환된 바이트 스트림의 크기가 메모리 buffer보다 클 경우, 노드 간 데이터 전송 과정에서 메모리 할당 과정이 추가로 발생하여 데이터 처리 시간이 증가하고, 전체적인 Spark의 성능 저하로 이어질 수 있다.

이에 본 논문에서는 Spark 환경에서 노드 간 데이터 전송 시 직렬화 과정에서 발생하는 메모리 할당을 최적화하기 위한 시스템을 제안한다. 제안하는 시스템은 Spark 환경에서 데이터를 처리하기 전 단일 애플리케이션을 통해 전체 데이터의 직렬화된 크기를 구하고 전체 데이터를 Executor로 나눠 최종 직렬화 데이터의 크기를 예측한다. 그 후 Spark 직렬화 파라미터 설정을 통해 메모리 할당을 진행한다.

본 논문의 구성은 다음과 같다. 2장에서는 빅데이터 처리 플랫폼 spark와 데이터 직렬화, Kryo를 알아보고 3장에서는 본 논문에서 제안하는 Spark 환경에서 처리 성능 향상을 위한 직렬화 buffer 최적화 시스템을 알아보고 4장에서는 결론 제시한다.

2. 관련 연구

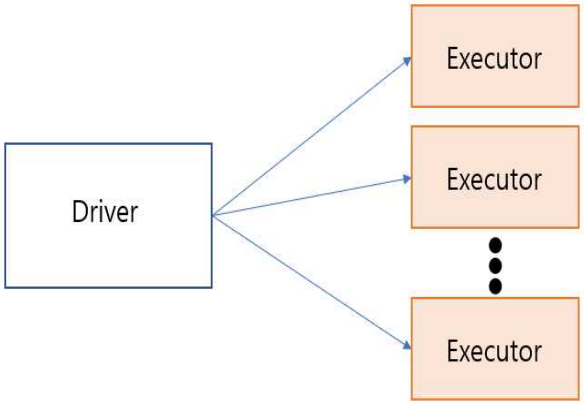
본 장에서는 빅데이터 처리 플랫폼 spark와 데이터 직렬화, Kryo에 대한 관련 연구를 살펴본다.

2.1 Spark Application

Apache Spark는 대규모 데이터 처리를 위한

In-Memory기반 프레임워크로 Spark Application를 통해 처리 작업을 실행할 수 있다.[4] (그림 1)과 같이 Spark Application은 1개의 Driver와 N개의 Executor로 구성된다.

Driver는 한 개의 노드에서 실행되며, task 단위의 작업을 Executor에 전달하는 역할을 한다. Executor은 Driver가 할당한 작업을 수행하여 결과를 반환하는 역할을 한다.



(그림 1) spark application 아키텍처

2.2 데이터 직렬화/역직렬화

데이터 직렬화란 객체를 바이트 스트림으로 바꾸고 데이터 역직렬화는 데이터 처리를 위해 바이트 스트림을 다시 객체로 재구성하는 작업이다. 직렬화 작업은 바이트 스트림을 메모리 buffer에 저장하여 디스크에 저장하거나, 네트워크 통신에 사용할 수 있다.[2]

2.3 Spark에서 발생하는 직렬화/역직렬화 작업

Spark 환경에서 Driver가 Executor에게 작업을 할당하는 것은 노드 간의 데이터 전송이기 때문에 직렬화/역직렬화 작업이 필요하다. 직렬화 작업에서 얼마나 많은 바이트 크기가 나왔는지, 직렬화 작업이 얼만큼 걸리는지가 전체 분산 처리 효율에 지장을 준다.[4] 만약 노드들의 직렬화·역직렬화된 데이터를 담는 buffer의 크기가 부족하게 설정되어 있다면, 데이터를 담는 과정에서 overflow가 발생하고 buffer의 크기를 다시 조정한다.

2.4 Kryo 직렬화 라이브러리

Spark의 서드파티 라이브러리인 Kryo는 Java 내장 라이브러리와 같은 바이너리 형식의 직렬화 방식을 사용한다.[3] Kryo는 Java 내장 라이브러리보다 직렬화 작업에서 압축률이 좋기 때문에 바이트 크기가 작고 작업 시간이 짧다.[5]

Spark 직렬화는 <표 1>에 있는 내부 옵션으로 직렬화 설정을 변경할 수 있는데, 직렬화 성능 향상을 위해 buffer 사이즈를 설정할 수 있다.

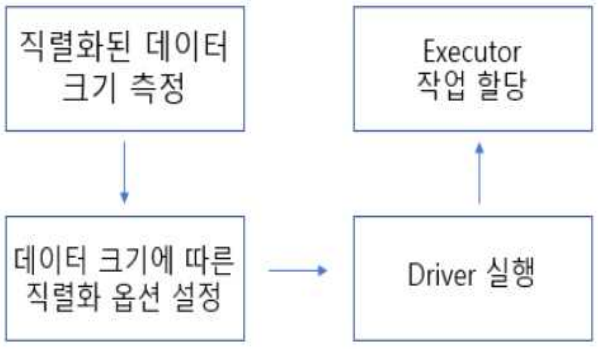
<표 1> Spark Serialization Parameter

파라미터	비고
spark.serializer	Spark 직렬화 라이브러리 설정
spark.serializer.buffer	직렬화 시 사용할 buffuer 크기 설정
spark.serializer.maxSize	직렬화된 데이터의 최대 크기를 설정하는 설정

3. 제안하는 Spark 환경에서 분산 처리 성능 향상을 위한 Buffer 최적화 시스템

본 장에서는 앞서 살펴본 관련 연구를 기반으로 Spark 환경에서 분산 처리 성능 향상을 위한 Buffer 최적화 시스템을 제안한다.

제안하는 방법 순서도는 (그림 2)와 같다. 기존 Driver에서 Executor에게 작업을 할당하기 전에 단일 애플리케이션 구현을 통해 데이터를 미리 직렬화하여 크기를 확인한다. 그리고 전체 데이터 크기를 Executor 수로 나누어 할당될 데이터 크기를 예측한다. 그 후 Spark 클러스터 노드들의 buffer 크기 설정 옵션을 변경하여 적절한 buffer를 할당한다.



(그림 2) 제안하는 Spark application 처리 순서

직렬화된 데이터 크기 측정 방법은 <표 2>와 같다. spark application에서 사용할 데이터를 object에 할당하고 kryo 라이브러리의 writeObject 메소드를

사용하면 kryo 직렬화 데이터가 생성된다. 그 후 total 메소드를 통해 최종 크기를 확인할 수 있다.

<표 2> kryo 직렬화 데이터 크기 측정 의사코드

```
1  Kryo kryo = new Kryo();
2  // 직렬화된 데이터 생성
3  Output ouput = new Output(new
4  ByteArrayOuputStream());
5  kryo.writeObject(output, object);
6  output.close();
7  // 직렬화된 데이터의 크기 계산
8  int size = ouput.total();
```

데이터 크기에 따른 직렬화 옵션 설정은 <표 3>과 같다. 첫 번째로 SparConf 설정을 가져온 후에 set 메소드를 통해 동적으로 직렬화 라이브러리를 설정할 수 있다. 두 번째로 <표 2>에서 나온 직렬화된 데이터 크기를 executor 개수로 나눠 최종 buffer 크기를 설정한다.

최종적으로 Driver가 execoutor에게 작업을 할당하기 전 직렬화된 데이터 크기를 측정하고 데이터 크기에 따른 직렬화 옵션 설정을 통해 적절한 buffer를 executor에게 할당할 수 있다.

<표 3> kryo 직렬화 옵션 설정 의사코드

```
1  SparkConf conf = new SparkConf();
2  // kryo 직렬화 사용
3  conf.set(“spark.serializer”, “org.apache.spark.
4  serializer.kryoSerializer”)
5  // 직렬화 데이터에 따른 buffer 설정
6  conf.set(“spark.serializer.buffer”,datasize/exe
7  cutorNum)
8  conf.set(“spark.serializer.maxSize”,1024)
```

4. 결론

Spark 클러스터 노드들은 직렬화·역직렬화 된 데이터를 담는 buffer의 크기가 부족하게 설정되어 있다면, 데이터를 담는 과정에서 overflow가 발생하고 buffer를 추가로 할당하는 과정이 발생한다. 이는 전체적인 Spark의 처리 성능을 낮추는 이유 중 하나이다.

이에 본 논문에서는 Spark 환경에서 처리 성능 향상을 위한 Buffer 최적화 시스템을 제안한다. 제안하는 방법은 직렬화된 데이터 크기 측정과 데이터 크기에 따른 옵션 설정을 통해 executor에 적절한 buffer를 할당해 처리 성능을 향상시킬 수 있다. <표 2>, <표 3>은 제안하는 방법의 직렬화된 데이터 크기 측정과 데이터 크기에 따른 옵션 설정 의사코드이다. 추후, 제안하는 방법의 검증을 위해 실제 Spark 클러스터 환경에서 성능 평가가 필요하다.

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2023-2020-0-01602)“

참고문헌

[1] Statista Research Department, Volum of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025

[2] Jaeyoung Jang, A Specialized Architecture for Object Serialization with Applications to Big Data Analytics, ISCA, 13.06.2020

[3] “Spark Kryo”, https://spark.apache.org/docs/latest/tuning.html

[4] “Spark”, https://spark.apache.org

[5] “Kryo”, https://github.com/EsotericSoftware/kryo