Spark 환경에서 네트워크 병목 현상을 예방하기 위한 클러스터 구성 방법 연구

홍석민[1], 유연준[1], 신용태[2]
[1]숭실대학교 컴퓨터학과
[2]숭실대학교 컴퓨터학부
ghdtjrals3@soongsil.ac.kr
dbduswns1997@soongsil.ac.kr
shin@ssu.ac.kr

A Study on Cluster Configuration Method to Prevent Network Bottleneck in Spark Enviroment

Seok-Min Hong[1],Yeon-Jun You[1], [2]Yong-Tae Shin
[1]Dept. of Computer Engineering, SoongSil University
[2]School of Computing, SoongSil University

요 약

Spark는 대용량의 데이터를 처리를 위해 분산된 데이터를 네트워크로 모은 다음, 데이터를 분할하는 작업인 Shuffle을 진행한다. 이때 Spark 클러스터의 어느 한 노드의 네트워크 전송 속도가 느릴 경우 병목 현상으로 인한 전체 처리 성능이 저하된다. 이에 본 논문에서는 네트워크 병목 현상을 예방하기 위한 클러스터 구성 방법을 제안한다. 본 논문에서 제안하는 노드 선택 시스템은 iperf 도구를 이용해 노드들의 대역폭을 측정하고 이에 따라 노드 선택 알고리즘을 통해 클러스터를 구성한다. 기존 Spark 클러스터와 본 논문이 제안하는 시스템으로 구성한 클러스터를 비교했을 때, 250MB 로그 파일을 제외하고 750MB 로그 파일부터는 네트워크 전송 속도가 낮은 노드를 가지고 있는 클러스터의 성능이 병목 현상으로 인해 느려졌다. 본 논문의 제안에 따라 노드들의 네트워크 전송 속도를 고려하여 클러스터를 구성하면 네트워크 전송 속도로 발생하는 병목 현상을 예방할 수 있다.

1. 서론

최근 매년 데이터의 크기가 커짐에 따라, 한정된 자원을 가지고 있는 기존의 단일 컴퓨터로 빅데이터를 처리하기 위해서는 오랜 시간이 걸린다. 반면에 분산 처리 시스템은 Scale-out 방식으로, 자원에 구애 받지 않고 시스템을 확장할 수 있기 때문에 현대에서 빅데이터를 처리하기 위한 분산 처리 시스템은 필수적이다.

Hadoop과 Spark는 가장 대표적인 분산 처리 프레임워크다. 그 중 Spark는 In-Memory 기반으로 Hadoop 보다 더욱 빠르기 때문에 많은 분야에서 사용되고 있다.[1]

Spark는 대용량의 데이터를 처리를 위해 분산된 데이터를 네트워크로 모은 다음, 데이터를 분할하는 작업인 Shuffle을 진행한다. 이때 분산 처리한 데이터를 네트워트를 통해 노드 간 데이터 전송이 발생하는데 네트워크를 통한 노드 간 데이터 전송은 Spark 클러스터의 어느 한 노드의 네트워크 전송 속도가 느릴 경우 병목 현상으로 인한 전체 처리 성능이 저하된다.[2]

이에 본 논문에서는 Spark 클러스터를 구성하는 각 노드들의 서로 다른 네트워크 성능로 인해 발생하는 네트워크 병목 현상을 해결하기 위해 노드들의 네트워크 전송 속도에 따른 클러스터 구성 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 분산 처리 프레임워크 spark, 노드의 네트워크 전송 속도를 측정할 수 있는 iperf에 대해 알아보고 3장에서는 spark 환경에서 노드들의 네트워크 전송 속도에 따른 분산 처리 클러스터 구성 방법을 제안한다. 4장에서는 3장에서 제안한 방법을 기반으로 기존의 분산 처리 클러스터 환경과 본 논문이 제안하는 방법의 클러스터 환경와 비교 분석한다. 마지막으로 5장

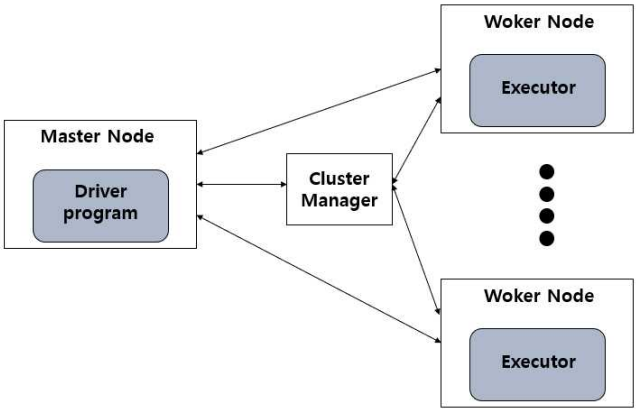
에서 결론을 제시한다.

2. 관련연구

본 장에서는 분산 처리 프레임워크 spark, 노드들의 네트워크 전송속도를 측정할 수 있는 iperf에 대한 관련 연구를 살펴보고 이를 통해 요구사항을 도출한다.

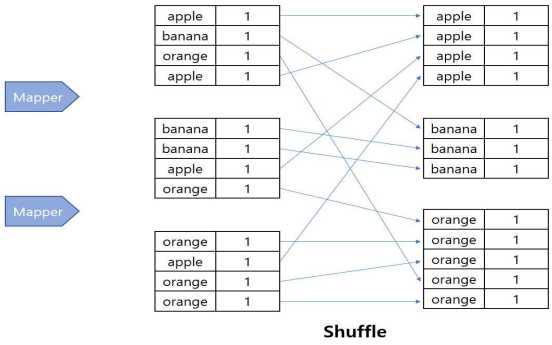
2.1 Apache Spark

Apache Spark는 In-Memory 기반의 프레임워크로 크기가 큰 데이터 처리를 수행할 수 있다. [그림 1]은 Spark Application의 아키텍처를 보여준다. Spark Application은 1개의 Driver와 N개의 Executors로 이루어져 있다.[3] Driver는 한 개의 노드에서 동작하며 task를 Executor로 전달하는 역할을 한다. Executor는 Driver로부터 할당 받은 task를 처리한 후 반환하는 역할을 수행한다.



[그림 1] Spark Application 아키텍처

이때 task를 할당받은 Worker 노드들은 처리 중 노드 간 데이터 전송이 필요할 때 Shuffle이란 작업을 통해 데이터를 전송한다. [그림 2]는 Apache Spark의 Shuffle 과정을 설명한다. Shuffle은 Key, Value의 데이터를 각 Partition으로 다시 배치하거나 재정렬하는 작업이다. 이때 네트워트를 통해 데이터를 전송하는데 만약 어느 한 Worker 노드의 처리율 혹은 네트워크 전송 속도가 느리다면 병목 현상이 발생하여 Spark 클러스터 전체 성능에 영향을 준다.[4] 본 논문에서 언급하는 병목 현상이란 컴퓨터 시스템에서 어떤 하나의 부분에 대한 성능이 떨어진다고 하면 컴퓨터 시스템의 전체 성능은 성능이 떨어지는 부분에 의해 전체적으로 성능이 저하되는 현상을 말한다.[5]



[그림 2] Spark의 Shuffle 과정

2.2 iperf

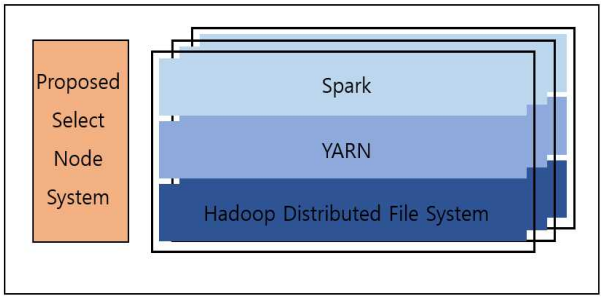
iperf는 C언로 이루어진 네트워크 성능 측정 도구이다. 클라이언트-서버 구조로 양방향간 네트워크 성능을 확인할 수 있기 때문에 분산 처리 시스템에서 Master 역할을 수행하는 노드가 여러 Worker Node로부터 네트워크 성능을 측정한 후 클러스터를 구성한다면 네트워크 전송 속도로 인한 병목 현상을 예방함으로써 네트워크 전송 속도로 인한 처리 성능 저하를 예방할 수 있다.[6]

3. 제안

본 장에서는 Spark 환경에서 노드들의 가용 네트워크 대역폭에 따른 클러스터 구성 방법을 제안한다.

3.1 제안하는 아키텍처

[그림 3]은 본 논문에서 제안하는 아키텍처이다. 제안하는 아키텍처는 노드 선택 시스템과 Spark 클러스터로 구성된다. 노드 선택 방법은 별도의 단일 애플리케이션으로 구현하며 iperf 도구를 이용하여 노드들의 가용 네트워크 성능을 측정한다. 그리고 노드 선택 알고리즘을 통해 최적의 노드들은 선택한다. Spark 클러스터는 본 논문에서 제안하는 노드 선택 알고리즘을 통해 구성한다.



[그림 3] 제안하는 아키텍처

3.2 제안하는 노드 선택 방법

제안하는 노드 선택 방법은 첫 번째로 iperf 도구를 활용해 노드의 네트워크 성능을 배열에 담는다. 두 번째는 처리할 데이터의 크기와 노드의 대역폭을 활용하여 각 노드별 처리 시간을 계산한다. 이때 클러스터의 전체 처리 시간(PT)은 Spark Worker 노드에 할당된 데이터(A)를 네트워크 전송 속도(S)로 나누어 시간(PT)을 계산하고 클러스터 구성하는 노드들의 경우의 수를 구한다. 클러스터 구성은 최소값이 나온 노드들의 집합으로 이루어진다. 수식은 다음 (1)과 같다.

$$PT = \max\left(N_{\left(\frac{A_1}{S_1}\right)}, N_{\left(\frac{A_2}{S_2}\right)}, \cdots, N_{\left(\frac{A_n}{S_n}\right)}\right) \quad (1)$$

노드들의 네트워크 전송 속도는 Master로부터 또 다른 task의 할당이나 다른 노드들의 네트워크 사용으로 인해 항상 일정하지 않다. <표1>, <표2>은 일정하지 않은 노드들의 네트워크 전송 속도를 주기적으로 요청하고 수집하여 기록하는 프로그램의 수도코드이다. 이를 통해 심각하게 네트워크 전송 속도가 낮아진 노드들을 확인하고 Spark 클러스터를 재구성 할 수 있다.

<표 1> 서버-클라이언트 네트워크 측정 결과 수집 프로그램 수도코드

```
1  # JSON 출력에서 bits_per_second 값을 추출
2  SPEED=$(echo $OUTPUT | jq
3  '.end.sum_received.bits_per_second')
4  # 추출된 값 로그 파일에 저장
5  echo "$(date): $SPEED bps“ >>$LOG_FILE
6  sleep 1
```

<표 2> 클라이언트-서버 네트워크 측정 요청 프로그램 수도코드

```
1  while true; do
2  iperf3 -c $MASTER_IP -t 60 -i 1 | grep
3  -E "^\\[ 4\\]" | awk '{print $7, $8}'
4  sleep 60
5  done
```

4. 성능평가

본 장에서는 기본 Spark 클러스터의 처리 성능과 본 논문에서 제안하는 방법으로 구성한 Spark 클러스터의 처리 성능을 비교 분석한다.

4.1 환경구성

본 논문에서 비교 분석을 위해 사용하는 노드들의 환경구성은 <표 3>과 같다.

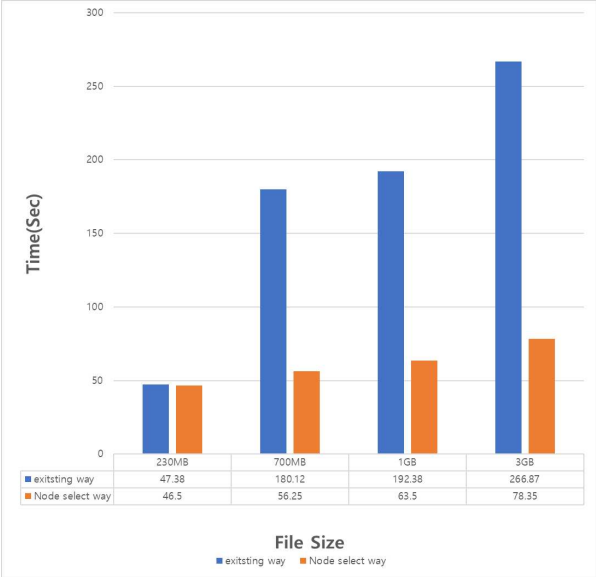
<표 3> 환경 구성 요소

	Master	Worker
Number of Node	2	5
CPU	2Core	1Core
MEMORY	8GB	4GB
DISK	HDD 50GB	HDD 50GB
OS	CentOS7	CentOS7
Spark	2.2.0	2.2.0

4.2 비교 분석

본 절에서는 다양한 대역폭 환경을 가지고 있는 노드들로부터 본 논문에서 제안하는 노드 선택 방법을 이용하여 구성한 클러스터와 기본 구성 클러스터와 비교 분석을 진행한다.

비교 분석에 사용한 파일은 WebServer 로그 파일로 250MB, 700MB, 1GB, 3GB 총 4가지 크기의 파일을 사용했다. Spark 내부 로직으로는 WebServer에 접근한 IP들을 분석한다. 각 크기의 파일마다 10번의 처리 시간의 평균을 계산하였다.



[그림 4] 기본 클러스터와 제안하는 방법으로 구성한 클러스터의 성능 비교

[그림 4]는 다양한 대역폭 환경을 가지고 있는 노드들로부터 본 논문에서 제안하는 노드 선택 방법을 이용하여 구성한 클러스터와 기본 구성 클러스터와 비교한 그래프이다. 250MB 파일에서의 처리시간은 파일의 크기가 크지 않기 때문에 대역폭에 크게 영향을 받지 않는다. 하지만 700MB 이후로 낮은 대역폭을 가지고 있는 클러스터의 Shuffle로 인한 데이터 전송 시간이 길어지면서 처리 시간이 크게 증가했다. 낮은 대역폭의 노드는 네트워크 병목 현상을 일으키기 때문에 클러스터 내에서 노드 간 데이터 전송이 잦을 경우 클러스터의 전체 성능이 크게 저하한다. 따라서 대역폭 환경에 따라 클러스터를 구성하는 노드들을 적절하게 선택해야 한다. 추가로 노드들의 가용 대역폭은 계속해서 변하기 때문에 지속적인 모니터링 기능도 필요하다.

5. 결론

Spark 환경에서 task를 할당 받는 노드들은 처리 중 노드 간 데이터 전송이 필요할 때 Shuffle이란 작업을 통해 데이터를 전송한다. 이 때 네트워크를 통해 데이터를 전송하는데 만약 어느 한 노드의 처리율 혹은 네트워크 전송 속도가 느리다면 분산 처리 시스템 전체 성능에 영향을 준다. 이에 본 논문에서는 iperf를 이용한 노드들의 네트워크 대역폭 측정과 노드 선택 알고리즘을 통해 Spark 환경에서 네트워크 병목 현상을 예방하기 위한 클러스터 구성 방법을 제안한다.

성능평가 결과는 노드 간 데이터 전송이 많아지는 큰 용량의 파일에서 낮은 대역폭의 노드로 인해 클러스터 처리 시간이 크게 늘어난 것을 확인 할 수 있었고, 반면에 본 논문에서 제안하는 노드 선택 방법을 이용하여 구성한 클러스터는 기존의 클러스터에 비해 낮은 처리 시간이 걸렸다.

추후 더 정확한 검증을 위해 특정 상황이 아닌 실제 동작 중인 대규모 클러스터 환경에서의 성능 평가가 필요하다.

“본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2023년도 문화기술 연구개발 사업으로 수행되었음 (과제명 : OTT 콘텐츠 저작권 보호 기술 개발 및 적용을 위한 저작권기술(+법) 융합인재양성, 과제번호 : RS-2023-00225267)“

참고문헌

[1] D. D. Mishra, S. Pathan and C. Murthy, "Apache Spark Based Analytics of Squid Proxy Logs," 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Indore, India, 2018, pp. 1-6, doi: 10.1109/ANTS.2018.8710044.

[2]Weiwei Gao, Xiaofeng Li and Dong Li, "Research on fixed traffic bottleneck of K-means clustering based on Hadoop," 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), Harbin, 2015, pp. 351-354, doi: 10.1109/ICCSNT.2015.7490767.

[3]https://spark.apache.org/docs/latest/cluster-overview.html

[4] https://spark.apache.org/docs/latest/tuning.html

[5] 한국정보통신기술협회,TTAS.KO-10.0258,정보시스템 성능관리 지침,2007년

[6]O. Olvera-Irigoyen, A. Kortebi and L. Toutain, "Available Bandwidth Probing for path selection in heterogeneous home Networks," 2012 IEEE Globecom Workshops, Anaheim, CA, USA, 2012, pp. 492-497, doi: 10.1109/GLOCOMW.2012.6477622.