누리온 시스템에서의 All-Reduce 알고리즘 성능평가

명훈주, 정기문
한국과학기술정보연구원
hjmyung@kisti.re.kr, kmjeong@kisti.re.kr

Performance Evaluation of All-Reduce Algorithms on Nurion System

Hunjoo Myung, Kimoon Jeong
Korea Institute of Science and Technology Information

요 약

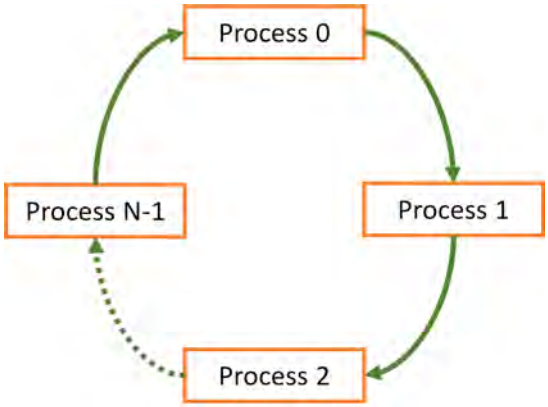
GPU 기술과 빅데이터의 성장에 힘입어 최근 딥러닝 기술은 괄목할만한 성장을 이루었고, 구글, 페이스북, 우버 등의 빅데이터를 보유한 업체들과 슈퍼컴퓨팅분야에서는 이러한 빅데이터를 빠른 시간 안에 학습하기 위해 분산 딥러닝 기술을 연구해오고 있다. 이러한 대규모 분산 딥러닝에서는 집합 통신, IO 부하 등이 주요 병목으로 알려져 있다. 본 연구에서는 분산 딥러닝에서 시도되고 있는 주요 All-Reduce 알고리즘들에 대해 누리온 시스템에서 성능평가를 수행하였고, 512노드 이상의 대규모에서는 2D-torus 알고리즘이 우수한 성능을 보였다.

1. 서론

GPU가 딥러닝 학습에 이용할 수 있고, CPU보다 훨씬 더 빠른 시간 안에 결과를 도출할 수 있다는 발표로 딥러닝이 재조명을 받기 시작한 시절에도, 대규모 분산 학습 딥러닝에 대해서는 회의적인 시각이 많았다. 클러스터 시스템이나 슈퍼컴퓨터에서 동작하는 응용프로그램은 일반적으로 프로그램 수행 중에 집합통신의 사용 비중이 높아질수록 병렬확정성은 떨어지기 때문이다. 딥러닝 학습은 순전파(forward)/역전파(backward)의 무수한 반복과정으로 말할 수 있는데, 분산 딥러닝 학습에서는 역전파 단계에서 각 노드가 계산한 경사값을 모두 합산해야 한다. 이때 All-Reduce 통신이 사용된다. 이것이 분산 딥러닝의 병렬확장성의 주요 병목이 된다. 본 논문에서는 이러한 병목을 극복하기 위해 제안된 여러 AllReduce 알고리즘들을 누리온 시스템에서 성능평가하고, 적용 전략을 고찰해본다.

2. 딥러닝에서의 집합 통신

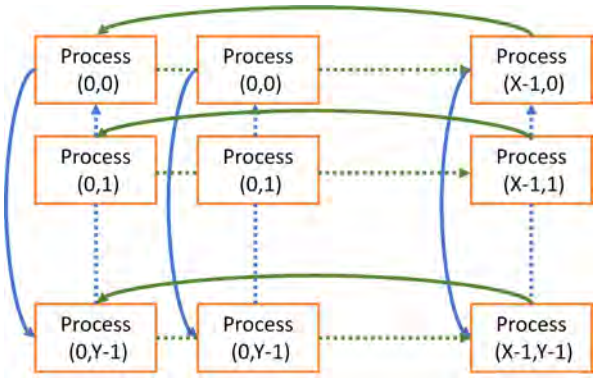
2017년 이후, 바이두[1]와 우버[2] 등에서 Ring-AllReduce 알고리즘을 분산 딥러닝 학습에 적용함으로써 대규모 분산 딥러닝 학습의 가능성을 열어 보였다. Ring-AllReduce 알고리즘은 <그림1>과 같이 분산 딥러닝 학습에 참여하는 모든 계산 노드들을 링 (Ring) 토플로지로 구성하고, AllReduce할 데이터는 노드 수만큼 쪼갠 후 하나씩 옆 계산 노드로 전달한다. 데이터를 넘겨받은 계산 노드는 자신의 데이터와 합산해 다시 옆으로 전달한다. 이러한 과정을 2(N-1) 번 (N : 계산노드 수) 반복하면 AllReduce가 완성된다. Ring-Allreduce의 장점은 참여하는 계산노드가 증가하더라도 통신비용은 N과 관계없이 데이터 크기에만 영향을 받는다.



<그림 1> Ring－구조

한편, 2018년 일본 ABCI에서는 Ring-AllReduce 알고리즘을 사용하는 대신에, 2D-Torus AllReduce 알고리즘[3]을 채택하여 성능향상을 꾀하였다. 2D-Torus AllReduce 알고리즘은 분산 딥러닝 학습에 참여하는 계산노드들을 2D-Torus 구성하고, 다음과 같이 3단계로 AllReduce를 수행한다.

① 수평방향으로 Scatter-Reduce를 수행한다.
② 수직방향으로 Ring-AllReuce를 수행한다.
③ 수평방향으로 All-Gather를 수행한다.



<그림 2> 2D-Torus 구조

Ring-AllReduce 알고리즘은 O(N)의 통신 횟수가 발생하는 한편, 2D-Torus AllReduce의 경우에는 O($\sqrt{N}$)의 통신 횟수를 가지는 터라 보다 효율적이다.

3. 누리온 시스템에서의 성능 평가

본 연구에서는 누리온 시스템에서 Ring-AllReduce 와 2D-torus AllReduce 를 중심으로 성능평가를 수행하였다. AllReduce 통신 자체의 성능보다는 통신 알고리즘이 딥러닝 학습에 얼마나 영향을 미치는지 알고 싶었기 때문에, 분산 딥러닝 벤치마크 코드 [4]를 수정하여 사용하였다. 딥러닝 소프트웨어로는 인텔의 OneCCL [5]과 연동한 pytorch를 사용하였다. OneCCL은 집합통신 라이브러리 중의 하나로, 여러 집합 통신 알고리즘을 제공하며, 환경변수 설정만으로 쉽게 알고리즘을 선택할 수 있다.

3.1. 실험환경

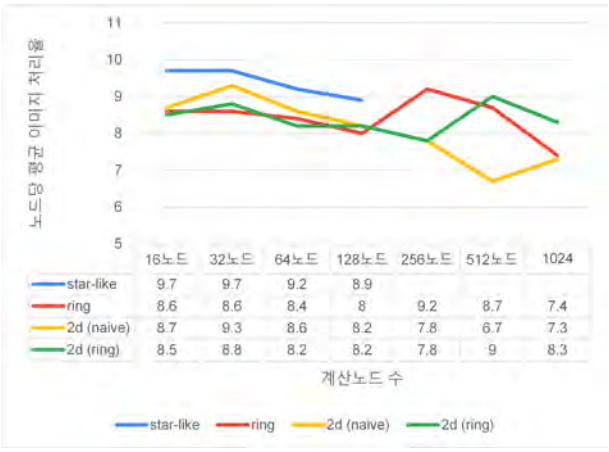
○ 시스템 : 누리온
- 계산노드 : Intel Xeon Phi(KNL) 8305 노드 +Intel Xeon Skylake 132 노드
- 인터커넥트 네트워크 : Intel OPA (FAT 트리 구조, 50% 블럭킹)
- 스토리지 : lustre 파일시스템 (20PB, 300GB/s), burst buffer (0.8PB, 800GB/s)

○ 소프트웨어:
- Pytorch (torch-ccl+OneCCL 연동) [6]

3.2. 실험평가

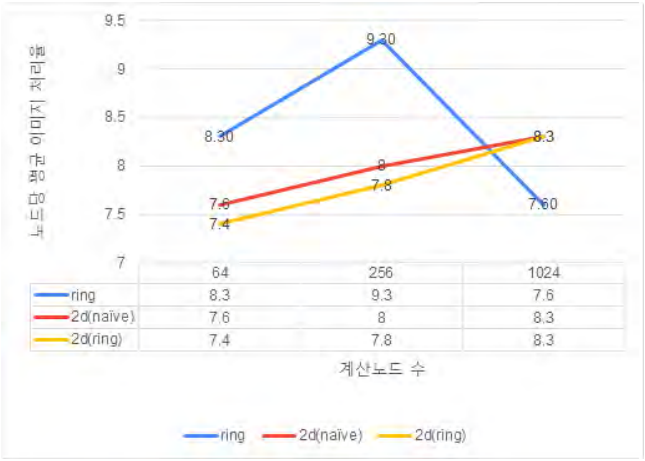
실험은 KNL 노드에서 1024노드까지의 병렬확정성을 평가하였고, 집합통신 성능에 보다 초점을 맞추기 위해 분산 딥러닝 벤치마크 수행에 있어서는 실데이터를 사용하지 않고 합성데이터를 사용하였다. 딥러닝 모델은 resnet-50을 사용하였고, 배치크기는 노드 당 8로 지정하였다.

<그림 3>은 AllReduce 알고리즘들의 노드 당 평균 이미지 처리율을 나타낸 것이다. 16노드에서 128노드까지는 starlike 알고리즘이 가장 우수한 성능을 보이고 있고, 256노드에서는 ring 알고리즘이, 512노드 이상에서는 2d-torus(ring) 알고리즘－세번째 단계인 all-gather를 ring 알고리즘을 사용－이 우수한 성능을 보였다. 즉, 성능을 평가한 AllReduce 알고리즘들은 노드가 증가함에 따라 각기 다른 성능을 보이는데, 2D-torus(ring)의 경우에는 대규모 분산 환경에서 우수한 성능을 보인다.



<그림 3> AllReduce 알고리즘 별 성능평가

<그림 4>은 2D-torus 알고리즘의 위상을 N*N 형태로 하여 성능 평가한 결과를 나타낸 것이다. 계산노드는 64 (8*8)노드, 256 (16*16)노드, 1024 (32*32)노드로 증가하면서 노드 당 평균 이미지 처리율을 측정하였다. 256노드 이하에서는 ring 알고리즘이 우수한 성능을 보였으나, 1024노드에서는 2D-torus 알고리즘이 좋은 성능을 보였고, 특히 2D-torus(ring) 알고리즘의 추세로 보아 1024노드 이상에서 세 가지 알고리즘 중에서 좋은 성능을 얻을 것으로 기대할 수 있다.



<그림 4> 2D-torus 알고리즘(N*N) 의 성능평가

4. 결론

본 연구에서는 대규모 분산 딥러닝 수행을 위해 시도되고 있는 AllReduce 알고리즘들을 살펴보고, 이 알고리즘들을 누리온 시스템에서 성능평가를 하였다. 본 실험에서 확인한 바와 같이 2D-torus 알고리즘이 누리온 시스템의 경우 512노드 이상에서 좋은 성능을 보였다. 그러나, 2D-torus 알고리즘과 같이 대규모 노드에서 좋은 성능을 보이는 것들이 작은 규모에서도 좋은 성능을 보이는 것은 아니여서, 각 시스템에서 본 실험과 같은 벤치마크 실험을 통해 참고할 수 있는 가이드를 만들어 놓는 것이 중요할 것으로 보인다.

참고문헌

[1]https://github.com/baidu-research/baidu-allreduce
[2] https://arxiv.org/pdf/1802.05799.pdf
[3] https://arxiv.org/abs/1903.12650
[4]https://github.com/horovod/horovod/blob/master/examples/pytorch/pytorch_synthetic_benchmark.py
[5] https://github.com/oneapi-src/oneCCL
[6] https://github.com/intel/torch-ccl