슈퍼컴퓨터 기반 교통 혼잡 예측을 위한 딥러닝 성능최적화

이홍석 정희진
한국과학기술정보연구원 슈퍼컴퓨팅본부
hsyi@kisti.re.kr hjjung1974@gmail.com

Performance of Deep Learning for Traffic Flow Prediction Using Supercomputer

Hongsuk Yi Heejin Jung
Korea Institute of Science and Technology Information, Supercomputing Center

요 약

최근 딥러닝의 등장은 복잡한 교통 빅데이터의 패턴 예측을 성공적으로 가능하게 하였다. 본 논문에서는 대도시 교통혼잡 예측을 가속화하기 위해 딥러닝 시뮬레이션을 인텔 제온 파이 코프로세서에서 수행하였다. 실험한 컴퓨팅 디바이스는 61개의 매니코어와 512-비트 벡터 레지스터를 가지고 있다. 교통 데이터는 단정도 FP32 연산으로도 충분하기 때문에 이 제온 파이에서 성능최적화가 가능하다. 테스트한 프로그램은 딥러닝의 오토인코더 알고리즘이며 병렬 성능은 61개 코어까지 성능 향상을 보았다. 하이퍼스레딩을 사용한 경우 약 1.2 테라플롭스의 성능을 얻을 수 있었으며, 이 수치는 이론성능의 약 50%이다.

1. 서 론

최근 알파고 인공지능의 성공으로, 교통공학을 포함하여 대부분 과학기술분야에서 딥러닝 연구가 활발히 진행하고 있다[1]. 2016년 3월 알파고와 이세돌 9단의 바둑 게임 결과는 인공지능이 얼마나 실제생활에 영향을 줄 수 있을 지를 잘 보여주었다.

최근 데이터 크기가 매우 복잡해지고 커지면서, 데이터 특성 분석을 주로 주성분분석(PCA)을 이용해왔다. 하지만 최근의 빅데이터는 대부분 비지도학습을 요구한다. 딥러닝의 여러 가지 학습방법들은 이러한 비지도학습을 시뮬레이션 가능하게 해주었다. 딥러닝 성공 요인 중 하나인 사전학습 방법으로는 오토인코더(Autoencoder)와 RBM(Restricted Boltzmann Machine)있다. 이들 방법을 적용으로 고차원의 복잡한 빅데이터 특성 및 패턴을 정확히 추출하게 하였다[2]. 딥러닝 알고리즘은 은닉층 개수의 증가로 엄청난 양의 계산을 해야 한다. 단일 CPU에서 순차코드는 제한된 시간 안에 계산을 완료하기 어렵다. 실제로, 알파고 알고리즘은 약 1,000개 CPU와 200 여개의 GPU를 장착한 슈퍼컴퓨터에서 병렬계산을 수행하였다고 알려졌다[3].

본 논문에서는 딥러닝 프로그램의 병렬성능 향상을 위해서 최신 슈퍼컴퓨터 아키텍처인 인텔 제온 파이 코프로세서를[4] 사용하였다. 사용한 데이터는 대도시 GPS 프로브 차량 데이터이다[5]. 이 데이터를 위해 필요한 연산은 차량 속도를 계산하기 때문에 단정도 연산이면 충분하다. 도로별 속도의 군집화 패턴을 분석하기 위하여 오토인코더 알고리즘을 적용하였다. 인텔 제온 파이 디바이스를 위한 병렬화는 GPU의 CUDA와 유사하다. 즉, 호스트 메모리의 데이터를 PCIe 버스로 연결된 제온 파이 코프로세서 메모리로 복사를 해야 하는데, 이를 Offload라고 한다. 이 디바이스는 61개 코어가 같은 메모리 공간을 공유하고 있어서 OpenMP 병렬화가 필요하다. 시뮬레이션 결과 얻은 성능은 단정도연산을 사용하면 1.2 테라플롭스를 얻을 수 있다. 이 수치는 제온 파이 디바이스도 GPU처럼 딥러닝에 매우 효율적인 아키텍처임을 알 수 있다.

2. 연구내용 및 방법

2.1 서울시 강남구 GPS 차량궤적데이터

본 연구에서 사용된 GPS 차량궤적데이터는 도착시각기준 링크통행속도이며 차량에 내장된 GPS 단말기를 통해 수집하였다. 각 차량의 GPS 단말기에서 매 1초마다 위치정보가 생성된 후 단말기에 내장된 링크정보에 GPS 위치데이터를 지도 매칭시켜 차량이 통과한 링크통행시간을 산출한다. 이 링크통행속도는 차량이 링크를 완전히 통과한 시점에 생성되므로 도착시각기준 링크통행시간이다. 사용된 링크통행속도는 링크정보의 링크 길이를 도착시간기준 링크통행시간으로 나누어 산출한다.

사용된 자료는 서울 강남구에 있는 간선도로로서 국가교통정보센터를 통행 관리·배포되고 있는 국가표준 노드링크에 등록된 노드와 링크에서 2013년 4월 1일에서 2013년 9월 30일에 수집된 6개월 데이터이다. 강남구에는 1480여 개 링크가 국가표준 노드링크에 등록되어 있

다. 이중 GPS 차량궤적데이터가 수집된 논현로 북쪽(구룡삼거리에서 동호대교 남단 교차로까지) 링크는 총 길이 6.78km이고 33개 링크였다. 편도 3차로이고 제한속도는 60km/h이다. 연구대상지의 33개 링크에서 수집된 probe 차량의 링크통행 속도는 해당 링크의 소통상황을 반영한 데이터이다[6].

본 연구에서는 15분 주기로 각 링크를 통과한 probe 차량들의 평균통행속도를 Traffic Performance Index로 환산하여 정체도를 측정하였다. 2016년 4월1일에서 2016년 6월 30일까지 3개월 (공휴일을 제외한 88일) 간의 정체도를 측정하여 주성분분석을 수행하였다. 각 링크의 통행특성은 하루 기준으로 96개의 정체도 패턴으로 정의하여 패턴의 유사성으로 링크 간 특성을 분류하였다.

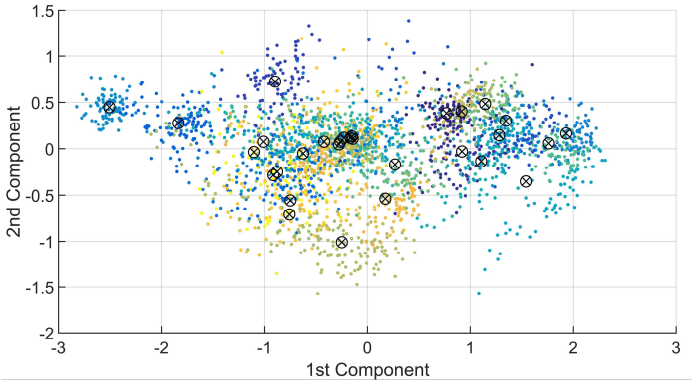

그림 1 링크 통행특성의 분류를 위한 주성분(Principal Component Analysis) 분석 결과

주성분 분석 결과 제1성분이 약 59%, 제 2성분이 약 10%의 해석력을 가지고 있었으며 레이블 된 링크 아이디 정보와 함께 비교한 결과 [그림 1]과 같은 분포를 2차원 주성분 공간에서 보였다. 여기서 Ⓧ표식은 각 링크별 통행패턴의 중심점으로 중심점간 거리에 따라 링크 간 유사성을 분류할 수 있었다.

2.2 딥러닝 오코인코더 알고리즘

2000년 초에 딥러닝 성공의 중요한 요인으로 심층 신경망을 사전학습 할 수 있는 오토인코더[7]의 등장이다. 그림2는 10개와 1개의 뉴런을 갖는 2개의 은닉층이 대칭구조로 된 신경망이다. 은닉층의 활성함수는 입력데이터의 특성에 따라서 선형함수 혹은 비선형 함수를 사용한다. 오토인코더 신경망은 비지도학습의 일종이고 주성분분석과 거의 유사한 역할을 한다.

입력층에서 출력층까지 신경망의 웨이트 값은 역전파 알고리즘을 사용하며, 목적함수는 기대치와 입력치의 오차를 최소화 하는 방법을 사용하며, 오차는 두 변수 차이의 제곱으로 계산되며 보통은 stochastic gradient descent 방법으로 계산된다. 오토인코더 기법은 비지도학습을 사전학습을 통해서 특성을 뽑아낼 수 있기 때문에 딥러닝 성공에 매우 중요한 역할을 하였다. 왜냐하면 라지-스케일 교통데이터처럼 모든 데이터에 라벨을 정하는 것은 많은 비용이 들며, 실제로 거의 불가능하기 때문에 비지도학습 방법을 사용해야한다.

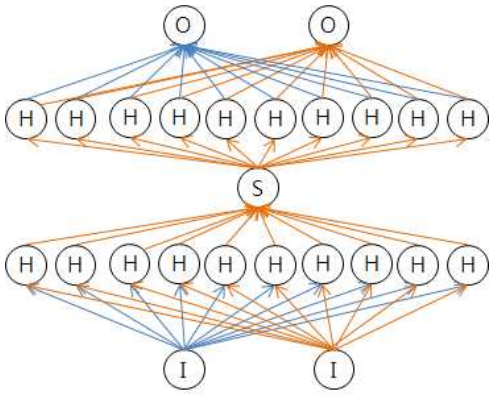

그림 2 딥러닝 오토인코더 신경망 모형

3 인텔 제온 파이에서 딥러닝의 병렬 성능

그림3은 인텔 제온 7120 한 개를 사용할 경우 스레드 개수의 증가에 따른 성능을 나타낸 것이다. 선형 주성분분석 PCA 코드를 사용하였고, 제온 파이를 이용한 병렬화는 호스트 메모리의 데이터를 제온 파이 메모리로 명시적인 복사 명령어를 사용하는 오프로드(Offload) 기법을 적용하였다. 딥러닝 알고리즘의 병렬화는 데이터 병렬화 방법으로 했으며, 각 샘플을 독립적인 이벤트로 취급하면 각 프로세스 사이의 데이터 통신은 없이 거의 선형의 병렬성능 확장성을 얻을 수 있는 방법이다.

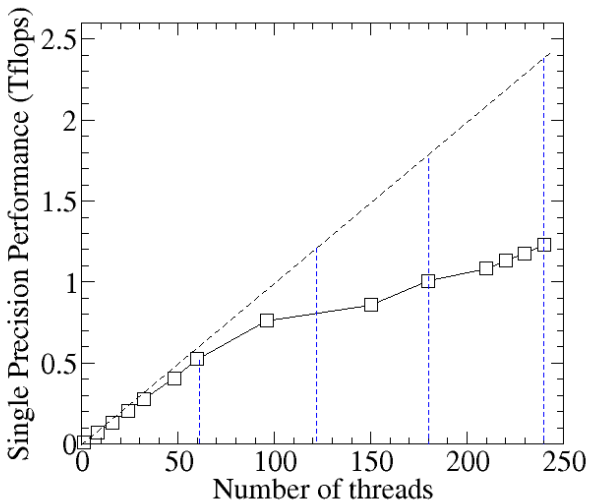

그림 3 인텔 제온 파이의 딥러닝 성능 확장성

공유메모리에서 OpenMP를 이용한 병렬화를 적용하였다. 사실 OpenMP의 pragma 지시어는 인텔 제온 파이 코프로세서에서도 쉽게 병렬화 루틴을 지원해준다. 하지면 여기서는 인텔에서 제공하고 있는 오프로드를 사용하고자 한다.

사용된 제온 파이 모델은 61개의 스레드 코어를 가지고 있으며 각각의 코어는 4개 까지 하이퍼스레딩을 할 수

있어서 총 244개의 스레드를 사용할 수 있다. 파이 아키텍처에서 주 메모리와 레지스터 사이의 대역폭은 352 GB/s이다.

```
#pragma offload_attribute (push, target(mic))
  #pargma omp parallel for reduction(+: sum)
    for(int index=0; index < Nsize; index++)
        sum += mean_square_error();
#pragma offload_attribute (pop)
```

그림 4 OpenMP 기반 Offload 프로그래밍 기법

이 디바이스는 1.238 GHz 코어를 가지고 있으며 각 코어는 512 비트 벡터 레지스터를 가지고 있어서 배정도 연산의 경우는 8배의 성능 향상을, 단정도 연산의 경우는 16배의 성능을 얻을 수 있다. 이 디바이스는 더셈과 곱셈을 동시에 지원하는 FMA를 제공하므로 총 성능은 61x1.238x8x2인 1.208 테라플롭스 성능을 가지며, 단정도 연산의 경우는 배정도 연산의 2배인 2.415 테라플롭스 이론 성능을 얻을 수 있다.

그림 3에서는 단정도 연산을 사용하였다. 성능확장성을 보면 실제 물리 코어인 61개 까지는 거의 선형적인 성능 향상을 볼 수 있다. 그리고 120개 정도 까지도 약간의 성능 향상을 보여주고 있지만 이는 하이퍼스레딩의 효과이기 때문에 선형적인 성능 향상을 얻을 수 없었다. 최고 얻을 수 있는 성능은 244 스레드를 사용할 경우 약 1.2 테라플롭스의 성능을 얻었다. 이 값은 이론값의 약 50%에 해당하는 수치이다. 보통 계산과학에서 단일 CPU에서 얻을 수 있는 성능은 보통 10%정도에 해당한다는 것을 기억한다면, 이 딥러닝 알고리즘을 이용하여 얻은 성능은 매우 우수하다고 말할 수 있다[8]. 즉, 딥러닝 알고리즘은 인텔 매니코어 파이 디바이스에서 최적의 성능을 얻을 수 있음을 의미한다.

4. 결론 및 향후 연구과제

본 논문에서는 인텔 제온 파이 코프로세서에서 서울시 강남구 지역 GPS 궤적차량 빅데이터를 분석하여 교통흐름 패턴을 확인하였다. 강남구에서는 오전 첨두 시간에 뚜렷한 교통흐름 정체 패턴을 확인하였다. 최근 빅데이터 기반 딥러닝 이론을 교통에 적용하는 방법을 제시하였다. 딥러닝 오토인코더를 이용한 교통 알고리즘은 계속 개발 중에 있으며, 비지도 학습과 사전학습을 통해서 얻은 결과를 이후 딥러닝의 RNN을 예측을 하면 매우 정확한 데이터를 통한 교통 혼잡을 예측할 수 있을 것이다. 또한 인텔 제온 파이 여러 대를 사용하기 위하여 MPI를 이용한 성능최적화를 진행하고 있다. 그리고 딥러닝 알고리즘을 최적화하기 위한 여러 가지 성능최적화 방법들을 적용할 계획이다.

5. 참고문헌

[1] Xiaolei Ma, Haiyang Yu, Yunpeng Wang, Yinhai Wang, Large-Scale Transportation Network Congestion Evolution Prediction Using Deep Learning Theory, PLOS ONE, 10(3), 2015.

[2] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.
Deep Learning Tutorials, DeepLearning 0.1 documentation http://deeplearning.net/tutorial/

[3] David Silver, et al. Mastering the game of Go with deep neural networks and tree search, Nature, 529, 484-489 (2016)

[4] J. Rendeirs, An Overview of Programming for Intel Xeon processors and Intel Xeon Phi coprocessors, 2015 Available: print

[5] 이홍석, 정희진, 배상훈. 라지-스케일 GPS 차량 궤적 빅데이터를 위한 딥러닝 연구, 한국정보과학회 2015년 동계학술발표회 논문집 10-12 (2015)

[6] 김태욱, 배상훈, 정희진, 차량궤적데이터를 활용한 도심부 간선도로의 돌발상황 검지, 한국 ITS 학회논문지, 제13권, 4호, 1, 2014

[7] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks.

[8] L. Jin, Z. Wang, R. Gu, C. Yuan and Y. Huang, "Training large scale deep neural networks on the intel xeon phi many-core coprocessor", IPDPS Workshops, pp. 1622-1630