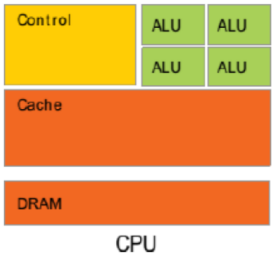
딥러닝 홀로서기#12

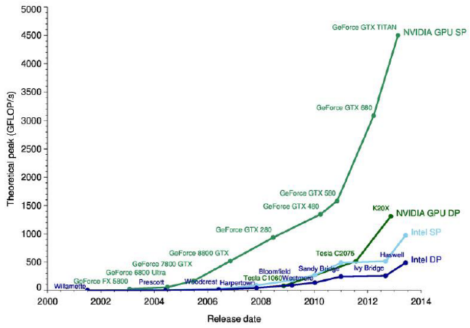
1. GPU

- GPU은 아키텍트가 KUDA Processor로 인하여 CPU보다 속도가 빠름
- GPU는 Matrix Operation을 연산하기에 최적화되도록 디자인되어 있음



CPU

GPU



2. How to use GPU with Pytorch?

- 다음과 같은 코드를 활용하여 모델을 CPU 또는 GPU에 로딩을 시켜준다
- .to(device)의 device란에 CPU인지 GPU인지 입력
- 단, 파이토치를 활용하여 GPU에 로딩이 가능한지 torch.cuda.is_available()로 확인을 먼저 해야함

```
# After define the model, and  then, use function of torch.device and model.to('CPU' or 'cuda:0'))
model = MLPModel(784, 10, [1000])
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

device = 'cuda:0' if torch.cuda.is_available() else 'cpu'
model.to(device)
```

- Data 또한 Model이 정의된 것과 동일한 곳에 로딩해주어야 함
- 여기서 Input_x값과 true_y값, loss값은 동일한 곳에 로딩 해주어야 함 (GPU 로딩이 빠름)
- 그렇지 않을 경우 런타임 Error가 발생가능
- GPU사용 시 loss계산시에도 numpy가 아닌, item() 함수를 활용해서 gradient detech연산과 cpu로딩, 그리고 numpy변환 연산을 한꺼번에 해줄 수 있다

```
input_x = input_x.to(device)
true_y = ture_y.to(device)
```

```
input_x = input_x.squeeze()
input_x = input_x.view(-1, 784)

pred_y = model(input_X)
```

※ 데이터 셋이 너무 작을 경우엔, CPU연산이나 GPU연산이나 그다지 큰 시간차이가 없으므로 굳이 GPU를 사용하지 않아도 된다.

※ 데이터 셋이 클 경우, GPU연산이 GPU연산보다 5배~20배까지도 빨라질 수 있다