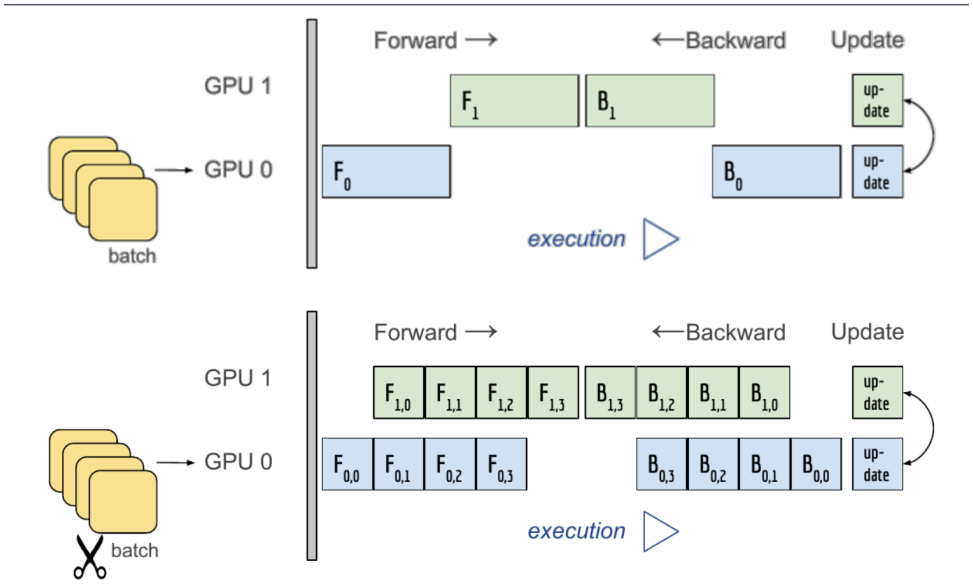
Multi-GPU 학습

- Single / Multi
- GPU / Node
- Single Node Single GPU
- Single Node Multi GPU
- Multi Node Multi GPU
- **Model parallel**

  다중 GPU에 학습 분산하는 방법: 모델/데이터 나누기

  모델 나누는 것은 예전부터 씀 (alexnet)

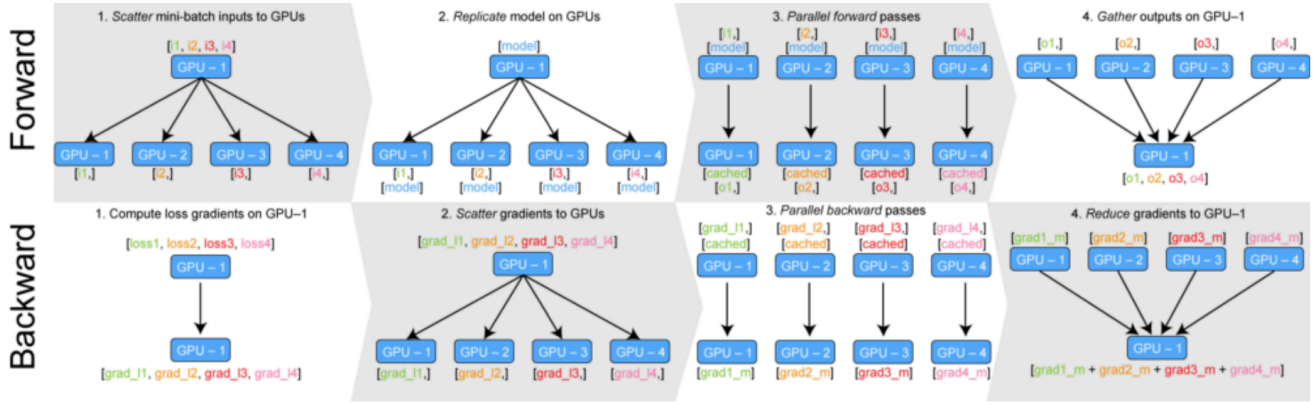
  모델의 병목, 파이프라인의 어려움 등으로 모델 병렬화, 고난이도 과제



- **Data parallel**

  데이터 나눠 GPU에 할당, 결과 평균 취함

  minibath 수식과 유사, 한 번에 여러 GPU에서 수행



  - **DataParallel**

    단순히 데이터 분배 후, 평균 취함

=> GPU 사용 불균형 문제 발생, batch 사이즈 감소(한 GPU 병목), GIL

- **DistributedDataParallel**

  각 CPU마다 process 생성, 개별 GPU에 할당

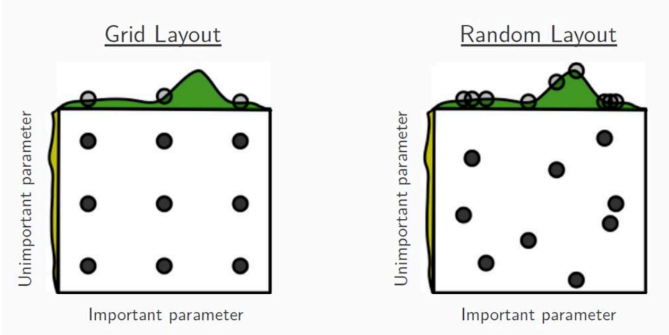
  => DataParallel로 하나 개별적으로 연산의 평균 냄

Hyperparameter Tuning

- 모델 스스로 학습하지 않는 값은 사람이 지정

  learning rate, 모델 크기, optimizer 등등

- 하이퍼파라미터에 의해 값 크게 좌우될 수도 있음 (요즘 별로)



- **Ray**

  ML/DL 병렬 처리 위해 개발된 모듈

  현재의 분산병렬 ML/DL 모듈 표준

  Hyperparameter Search를 위한 다양한 모듈 제공

PyTorch Troubleshooting

- 왜, 어디서 발생했는지 알기 어려움
- Error backtracking 이상한데 감
- 메모리 이전 상황의 파악 어려움
- **GPUUtil**

  nividia-smi처럼 GPU 상태 보여주는 모듈

  Colab은 환경에서 GPU 상태 보여주기 편함

  iter마다 메모리 늘어나는지 확인

- torch.cuda.empty_cache()

  사용되지 않은 GPU 상 cache 정리, 가용 메모리 확보

  del과는 구분 필요, reset 대신 쓰지 좋은 함수

- training loop에 tensor로 축척되는 변수 확인

  tensor로 처리된 변수, GPU 상에 메모리 사용

  해당 변수, loop 안에 연산 있을 때 GPU에 computational graph 생성(메모리 잠식)

    - 1-d tensor 같은 경우, python 기본 객체로 변환 처리
- del 명령어 적절히 사용
- 가능 batch 사이즈 실험 (1)
- torch.no_grad() 사용

  Inference 시점에서 사용

  backward pass로 인해 쌓이는 메모리에서 자유로움
- 예상치 못한 에러 메세지

  CUDNN_STATUS_NOT_INIT, device-side-assert, 등
- colab에서 너무 큰 사이즈 실행 X (linear, CNN, LSTM)
- CNN의 대부분 에러: 크기 안 맞아서 생김 (torchsummary 등 사이즈 맞춤)
- tensor의 float precision을 16bit 줄일 수도 있음