[DL] Distributed Training (분산 학습) 이란?

Distributed Training (분산 학습) 이란?

- 딥러닝 모델 설계 과정에는 많은 시간이 소요
- 따라서, 모델의 학습 과정을 가속화하는 것은 매우 중요
- **분산 학습은 이러한 딥러닝 모델의 학습 시간을 단축하는데 필수적인 기술 중 하나**

Distributed Training

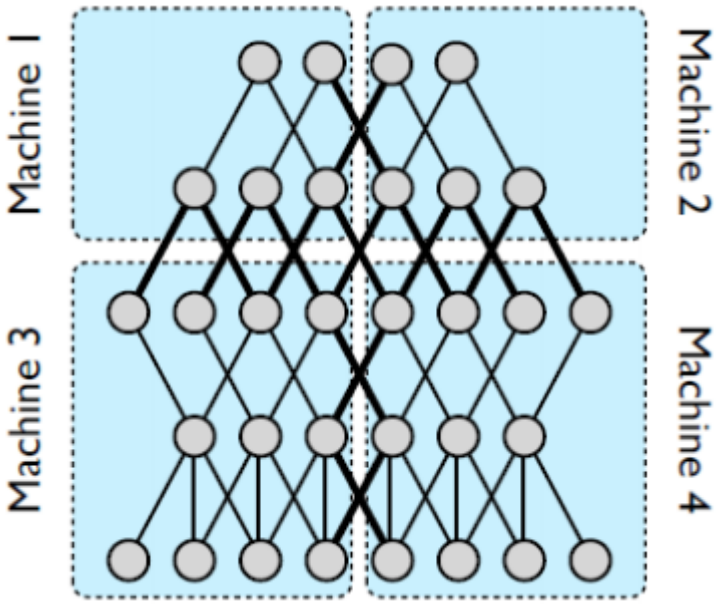
- 분산 학습의 핵심 개념은 크게 3가지로 분류
  - **작업 분할 방식**
    - Model Parallelism
    - Data Parallelism
  - **Parameter 동기화 방식**
    - Synchronous replication
    - Asynchronous replication
  - **Gradient 취합 방식**
    - All-Reduce (Parameter Server)

        - Ring-AllReduce

작업 분할 방식

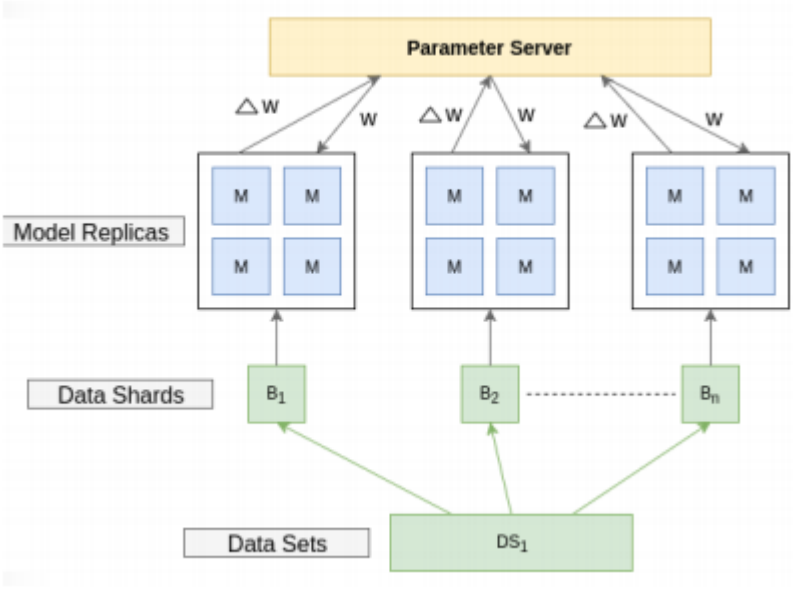
- **작업 분할 방식은 2가지로 분류**
    - **Model Parallelism**
        - 모델을 여러 GPU 에 분산하여 처리하는 방법
    - **Data Parallelism**
        - 데이터를 여러 GPU 에 분산하여 처리하는 방법

Model Parallelism



- **모델을 여러 GPU 에 분산하여 처리하는 방법이며, 입력 데이터는 동일**
- 즉, 전체 신경망 모델을 분리한 뒤, 각각의 모델을 여러 GPU 가 할당 받아 처리하는 방법
- 모델이 큰 경우에 사용 가능하며, 모델을 "잘" 분리하는게 관건

Data Parallelism



- **데이터를 여러 GPU 에 분산하여 처리하는 방법**
- 모델 학습은 다음과 같은 순서로 진행
  - 각각의 모델은 동일하며, 각기 다른 입력 데이터를 받아, 모델을 학습
  - 모델 학습 시, 순전파와 역전파를 통해, 모델의 파라미터가 결과값에 미치는 영향 (Gradient) 을 계산
  - 각각의 모델로부터 나온 Gradient 의 평균을 추출
  - Gradient 의 평균을 사용해, 모델의 파라미터를 업데이트

Parameter 동기화 방식

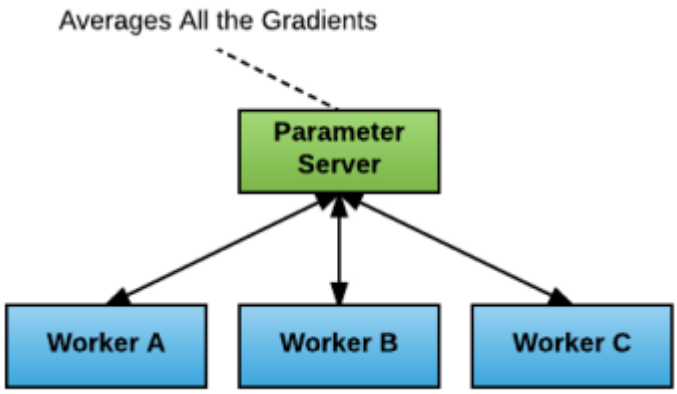
- **Parameter 동기화 방식은 2가지로 분류**
  - **Synchronous replication**
    - 분산처리가 모두 끝났을 때, Gradient 의 평균을 사용해 Parameter 를 업데이트
    - 수렴이 빠름
  - **Asynchronous replication**
    - 각 분산 처리에서 Gradient 계산이 끝난 건, 먼저 사용해서 Parameter 를 업데이트
    - 수렴이 느릴 가능성 존재

- Sync 방식은 모든 Worker(GPU)의 Job 이 끝날 때까지 대기했다가 Parameter 를 업데이트하므로분산 되어있는 Worker(GPU)가 많을 수록 Async 방식이 효율적

Gradient 취합 방식

- **Gradient 취합 방식은 크게 2가지로 분류**
  - **All-Reduce (Parameter Server)**
    - Parameter Server 가 모든 Gradient 를 취합하여 재분배
  - **Ring-AllReduce**
    - 모든 GPU 를 Ring 형태로 구성한 뒤, Gradient 전달을 통해 공유

All-Reduce (Parameter Server)

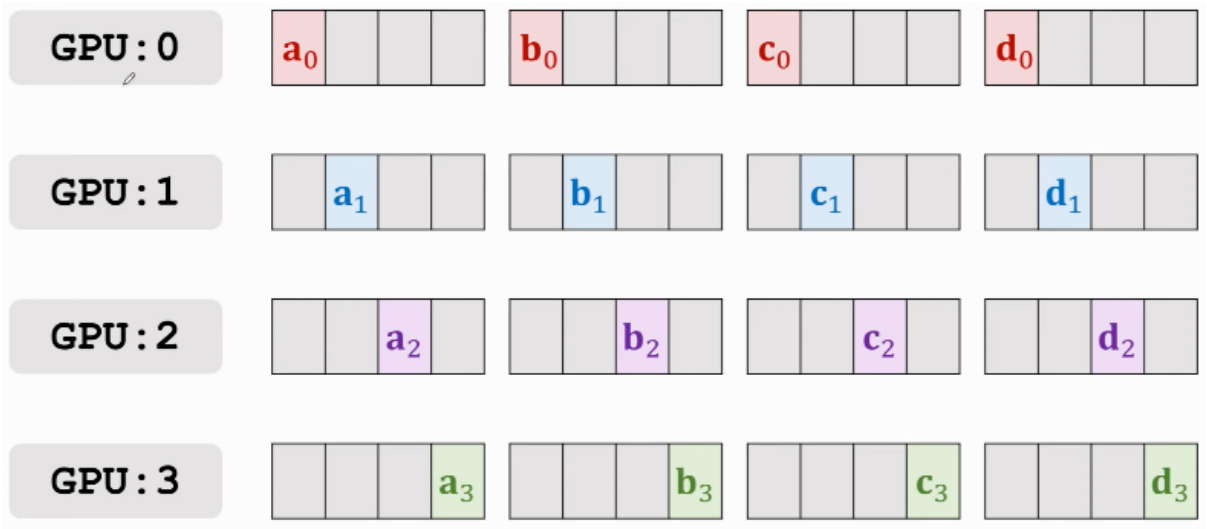


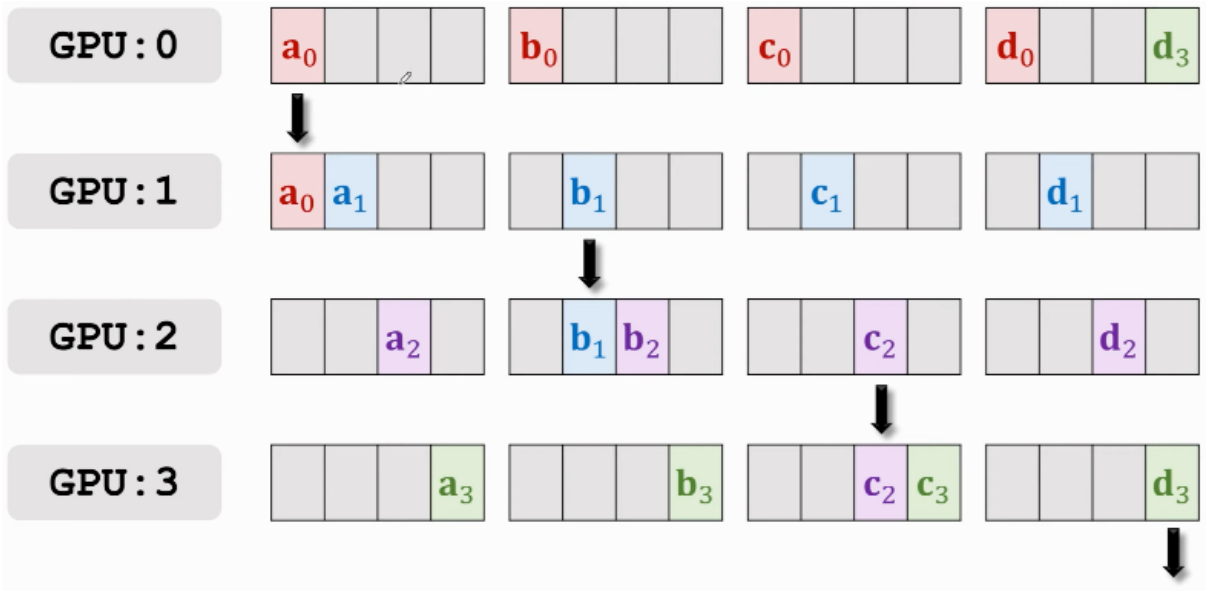
- All-Reduce 는 단순히, **Parameter Server 가 모든 Worker(GPU) 로부터 계산 된 Gradient 를 취합한 뒤, Worker(GPU) 에게 재분배하는 방식**
- 단순한 구조라는 것이 장점, Worker(GPU) 수에 따라, Parameter Server 의 메모리 사용량 및 네트워크 부하가 비례하여 증가하기 때문에 병목 현상이 발생할 수 있다는 점이 단점

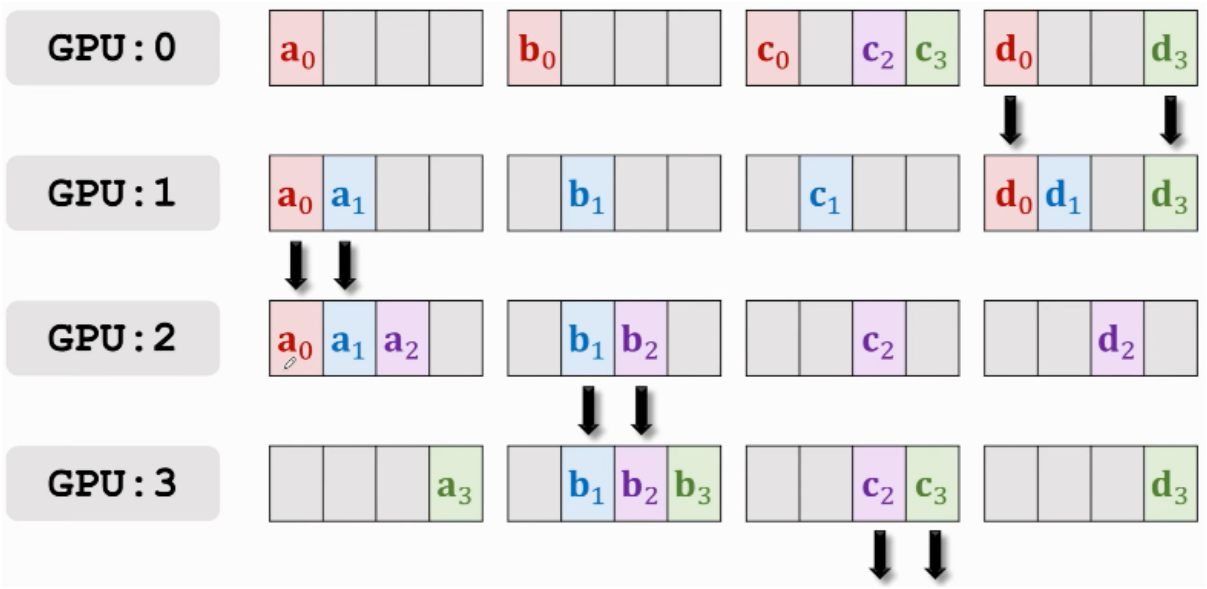
Ring-AllReduce

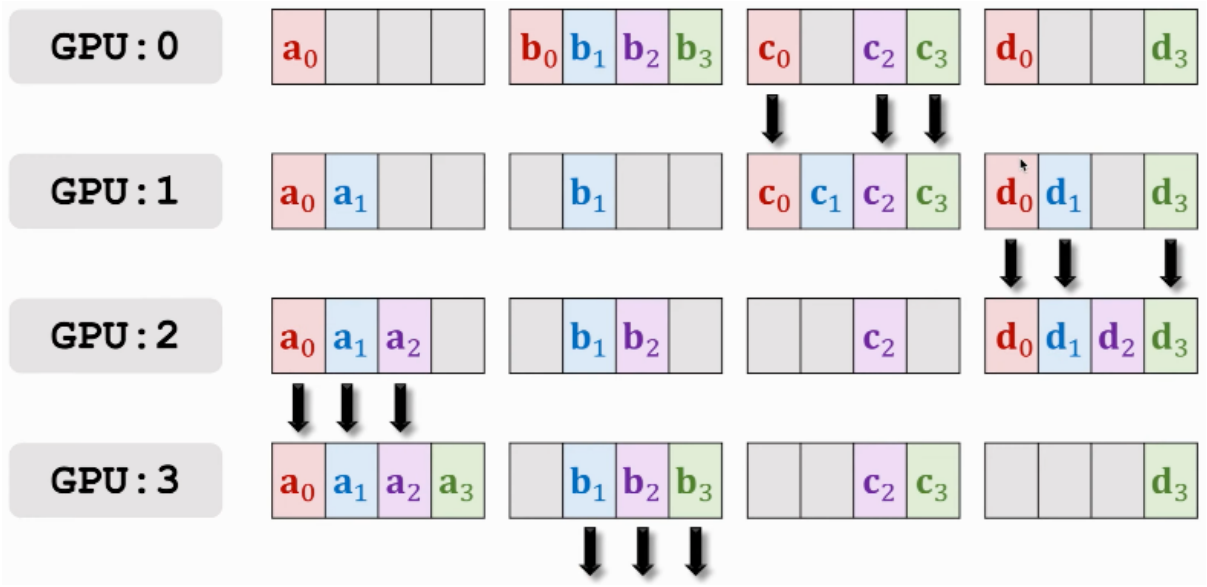
- **Ring-AllReduce 는 AllReduce 의 병목 현상을 제거한, 빠르고 확장 가능한 알고리즘 입니다.**
- **Ring-AllReduce 알고리즘의 순서는 아래와 같습니다.**
  1. 모든 GPU 를 Ring 형태로 구성합니다.
  2. GPU 에서 계산된 Gradient 를, 총 GPU 개수만큼 분리합니다.
  3. 각각의 GPU 는 분리된 Gradient 의 일부를, 옆 GPU 에게 전달합니다.
  4. Gradient 를 넘겨받은 GPU 는, 본인의 Gradient 와 전달 받은 Gradient 를 합산한 뒤, 다시 옆 GPU 에게 Gradient 를 전달합니다.
  5. 이러한 과정을 2(P-1) 번 진행합니다. [N : GPU 개수]
- 위의 그림을 예로 들었을 때,
- Start 그림에서 GPU0 의 a0 + b0 + c0 + d0 + e0 은, GPU0 에서 계산된 Gradient 를 의미합니다.
- 저희는 GPU0 ~ GPU4 가 가진 모든 Gradient 를 우측 Goal 그림처럼 취합하는게 목표입니다.

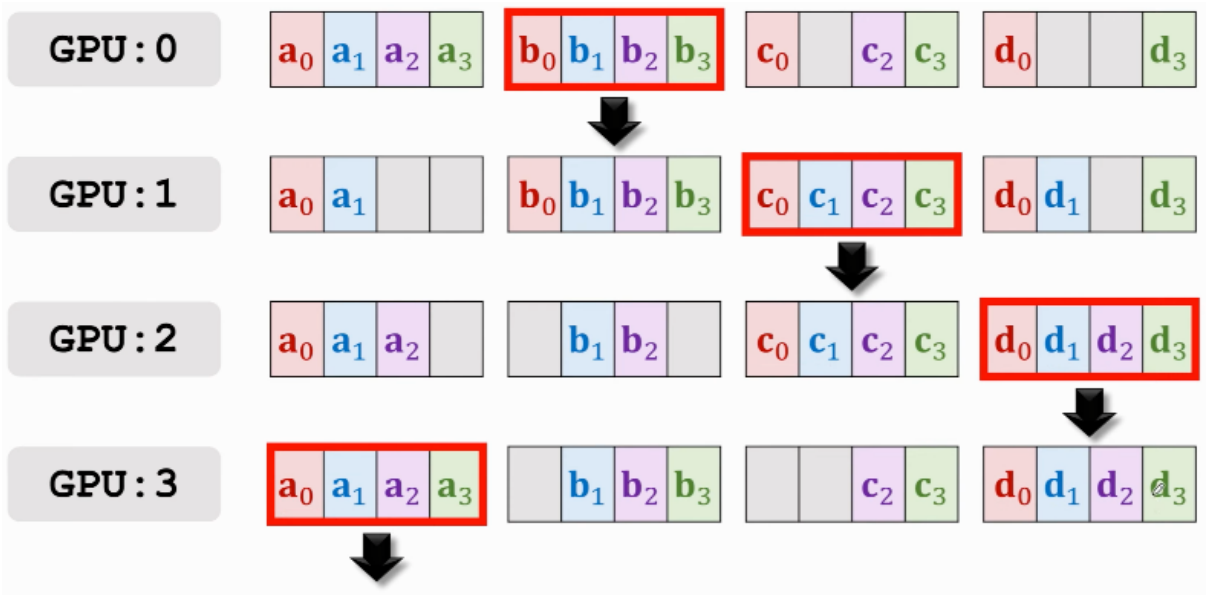
Ring-AllReduce 순서 예시

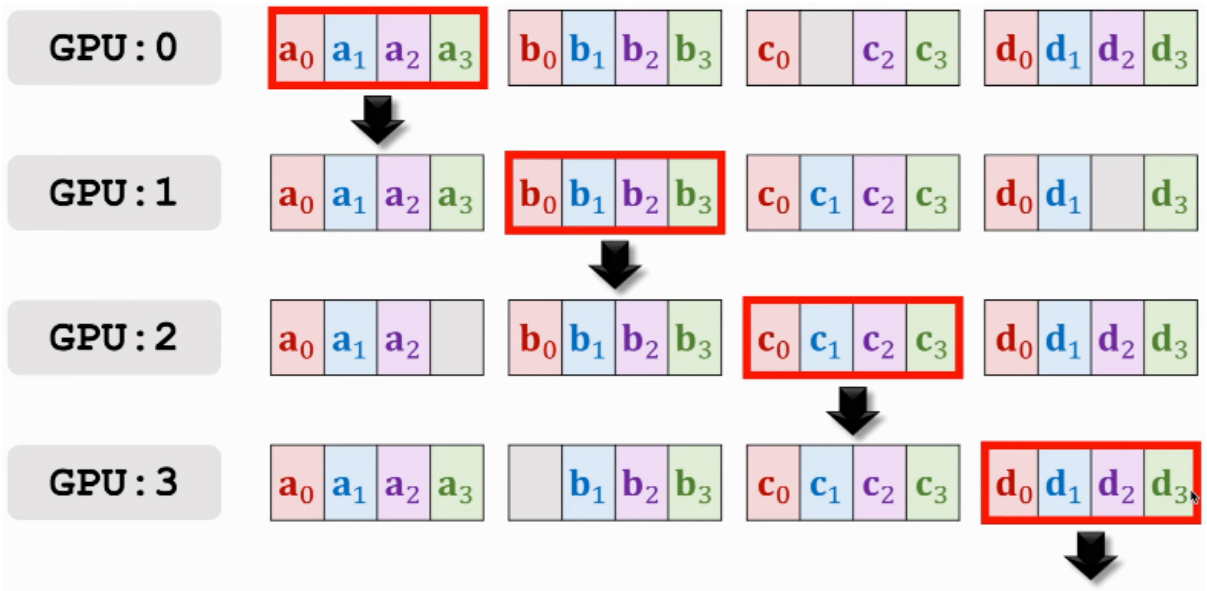

GPU : 0
GPU : 1
GPU : 2
GPU : 3

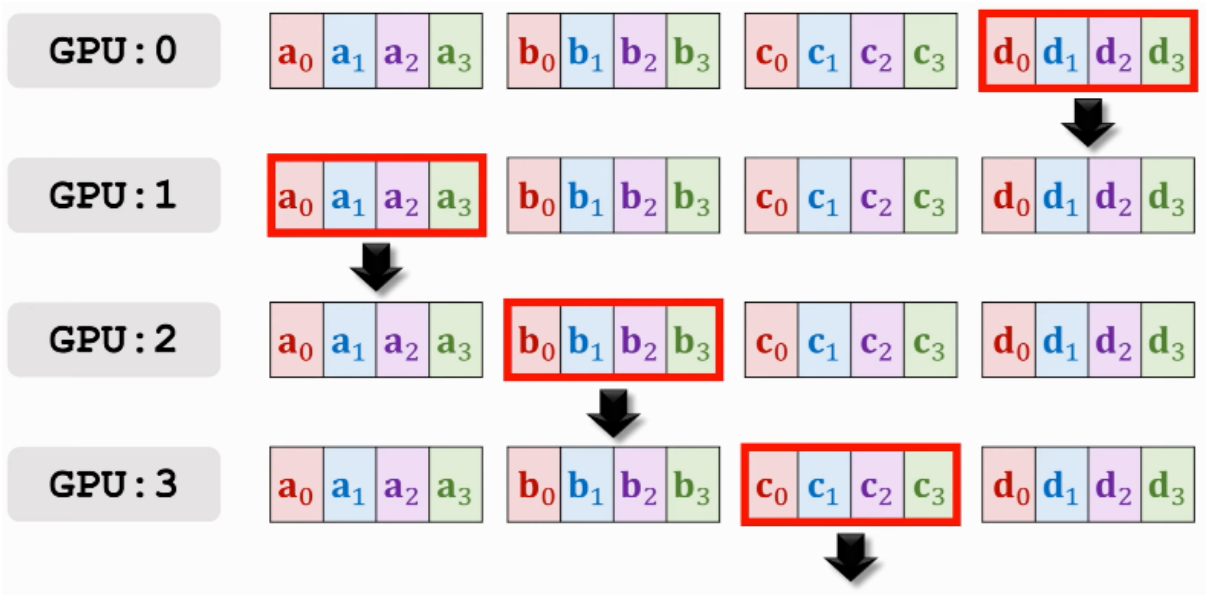

GPU : 0
GPU : 1
GPU : 2
GPU : 3


GPU : 0
GPU : 1
GPU : 2
GPU : 3


GPU : 0
GPU : 1
GPU : 2
GPU : 3


GPU : 0
GPU : 1
GPU : 2
GPU : 3


GPU : 0
GPU : 1
GPU : 2
GPU : 3


GPU : 0
GPU : 1
GPU : 2
GPU : 3
a_0 a_1 a_2 a_3
b_0 b_1 b_2 b_3
c_0 c_1 c_2 c_3
d_0 d_1 d_2 d_3