



(19) 대한민국특허청(KR)  
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0037849  
(43) 공개일자 2024년03월22일

(51) 국제특허분류(Int. Cl.)  
G06N 3/098 (2023.01) G06N 3/063 (2023.01)  
(52) CPC특허분류  
G06N 3/098 (2023.01)  
G06N 3/063 (2023.01)  
(21) 출원번호 10-2023-0118118  
(22) 출원일자 2023년09월06일  
심사청구일자 없음  
(30) 우선권주장  
1020220116393 2022년09월15일 대한민국(KR)

(71) 출원인  
삼성에스디에스 주식회사  
서울특별시 송파구 올림픽로35길 125 (신천동)  
울산과학기술원  
울산광역시 울주군 언양읍 유니스트길 50  
(72) 발명자  
신희안  
서울특별시 송파구 올림픽로35길 125 (신천동, 삼성SDS West Campus)  
하승훈  
서울특별시 송파구 올림픽로35길 125 (신천동, 삼성SDS West Campus)  
(뒷면에 계속)  
(74) 대리인  
특허법인씨엔에스

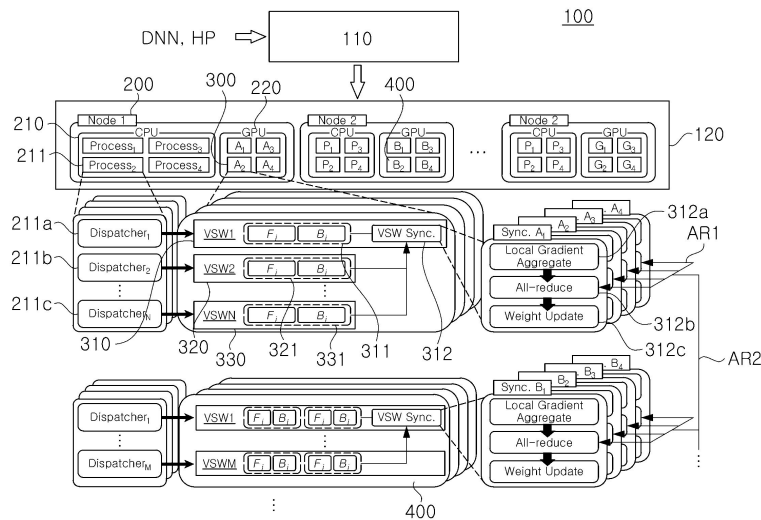
전체 청구항 수 : 총 18 항

(54) 발명의 명칭 이중 GPU들 기반의 분산 학습 장치 및 방법

(57) 요약

본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 장치에 있어서, 이중 GPU들 각각은 매 반복마다 멀티 스트림에서 실행되는 복수의 가상 스트림 작업자 - 복수의 가상 스트림 작업자 각각은 멀티 스트림 중 서로 다른 스트림에서 동시에 실행됨 - 별로 신경망 모델의 로컬 그래디언트를 연산하고, 복수의 가상 스트림 작업자 별로 구한 신경망 모델의 로컬 그래디언트에 기초하여 복수의 가상 스트림 작업자에 대해 신경망 모델의 파라미터를 동기화하며, 복수의 가상 스트림 작업자의 수는 성능에 따라 이중 GPU 별로 상이한 값을 가질 수 있다.

대표도



(72) 발명자

**최영리**

울산광역시 울주군 언양읍 유니스트길 50

**윤경찬**

울산광역시 울주군 언양읍 유니스트길 50

**엄상현**

울산광역시 울주군 언양읍 유니스트길 50

**정현준**

울산광역시 울주군 언양읍 유니스트길 50

## 명세서

### 청구범위

#### 청구항 1

이중 GPU들 기반의 분산 학습 장치에 있어서,

상기 이중 GPU들 각각은,

매 반복(iteration) 마다 멀티 스트림에서 실행되는 복수의 가상 스트림 작업자(Virtual Stream Worker, VSW)  
- 상기 복수의 가상 스트림 작업자 각각은 상기 멀티 스트림 중 서로 다른 스트림에서 동시에 실행됨 - 별로 신경망 모델(Neural Network Model)의 로컬 그래디언트를 연산하고,

복수의 가상 스트림 작업자 별로 구한 상기 신경망 모델의 로컬 그래디언트에 기초하여 복수의 가상 스트림 작업자에 대해 상기 신경망 모델의 파라미터를 동기화하며,

상기 복수의 가상 스트림 작업자의 수는, 성능에 따라 상기 이중 GPU 별로 상이한 값을 가지는, 분산 학습 장치.

#### 청구항 2

제1항에 있어서,

상기 복수의 가상 스트림 작업자는,

동일한 크기의 로컬 배치 사이즈( $B_L$ )의 데이터를 처리하는, 분산 학습 장치.

#### 청구항 3

제1항에 있어서,

상기 분산 학습 장치는, 복수의 노드를 포함하고,

각 노드는 동일한 성능의 GPU들을 포함하고, 각 노드에 포함된 GPU들은 다른 노드에 포함된 GPU들과는 다른 성능을 가지는, 분산 학습 장치.

#### 청구항 4

제3항에 있어서,

상기 이중 GPU들 각각은,

동일 GPU 내에서 상기 복수의 가상 스트림 작업자 중 메인 가상 스트림 작업자를 통해 복수의 가상 스트림 작업자 별로 구한 상기 신경망 모델의 로컬 그래디언트를 합산하여 집적된 그래디언트(aggregated gradient)를 구하고,

전체 축소 연산(All-reduce operation)을 통해, 상기 동일 GPU 내에서 구한 집적된 그래디언트, 상기 GPU와 동일한 노드에 속한 동종 GPU들에서 구한 집적된 그래디언트 및 상기 GPU와 상이한 노드에 속한 이중 GPU들에서 구한 집적된 그래디언트들을 평균하여 글로벌 그래디언트를 구하며,

상기 글로벌 그래디언트에 기초하여 상기 이중 GPU들 각각에 부여된 상기 신경망 모델의 파라미터를 업데이트함으로써 상기 신경망 모델의 파라미터를 동기화하는, 분산 학습 장치.

#### 청구항 5

제2항에 있어서,

상기 이중 GPU들 각각은,

GA(Gradient Accumulation) 기법을 이용하는 경우, GA 스텝수( $n_{GA}$ )( $n_{GA}$ : 2 이상의 자연수임)만큼의 로컬 그래디

언트를 누적하고, 누적된 로컬 그라디언트의 평균을 상기 로컬 그라디언트로 구하는, 분산 학습 장치.

#### 청구항 6

제1항에 있어서,

상기 이중 GPU들 각각은,

기 설정된 방식을 통해 복수의 가상 스트림 작업자들 중 메인 가상 스트림 작업자를 이용하여 나머지 가상 스트림 작업자들로 업데이트된 상기 신경망 모델의 파라미터를 복사하며,

상기 기 설정된 방식은,

상기 신경망 모델의 모든 레이어에 대한 전체 축소 연산(All-reduce operation)이 완료된 이후에 모든 레이어에 대한 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 순차 방식(sequential method); 및

상기 신경망 모델의 일부 레이어에 대한 전체 축소 연산이 완료되면 전체 축소 연산이 완료된 레이어에 대한 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 중첩 방식(overlapping method) 중 어느 하나를 포함하는, 분산 학습 장치.

#### 청구항 7

제5항에 있어서,

상기 분산 학습 장치는,

주어진 로컬 배치 사이즈에 따라 상기 복수의 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 상기 GA 스텝수( $n_{ga}$ )를 설정하는 설정부를 더 포함하는, 분산 학습 장치.

#### 청구항 8

제7항에 있어서,

상기 설정부는,

기 정해진 최대값 이하에서 성능이 최대가 되는 상기 복수의 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 상기 GA 스텝수( $n_{ga}$ )를 구하는, 분산 학습 장치.

#### 청구항 9

제7항에 있어서,

상기 설정부는,

상기 이중 GPU들 각각에 대하여, 다양한 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 다양한 GA 스텝수( $n_{ga}$ )의 조합으로 구성된 검색 공간을 구성하고,

상기 검색 공간 내에서 다이내믹 프로그래밍 기법을 이용하여 성능이 최대가 되는 상기 복수의 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 상기 GA 스텝수( $n_{ga}$ )를 구하는, 분산 학습 장치.

#### 청구항 10

이중 GPU들 기반의 분산 학습 방법에 있어서,

상기 이중 GPU들 각각에서, 매 반복(iteration) 마다 멀티 스트림에서 실행되는 복수의 가상 스트림 작업자(Virtual Stream Worker, VSW) - 상기 복수의 가상 스트림 작업자 각각은 상기 멀티 스트림 중 서로 다른 스트림에서 동시에 실행됨 - 별로 신경망 모델(Neural Network Model)의 로컬 그라디언트를 연산하는 단계; 및

상기 이중 GPU들 각각에서, 복수의 가상 스트림 작업자 별로 구한 상기 신경망 모델의 로컬 그라디언트에 기초하여 복수의 가상 스트림 작업자에 대해 상기 신경망 모델의 파라미터를 동기화하는 단계를 포함하며,

상기 복수의 가상 스트림 작업자의 수는, 성능에 따라 상기 이중 GPU 별로 상이한 값을 가지는, 분산 학습

방법.

#### 청구항 11

제10항에 있어서,

상기 복수의 가상 스트림 작업자는,

동일한 크기의 로컬 배치 사이즈( $B_L$ )의 데이터를 처리하는, 분산 학습 방법.

#### 청구항 12

제10항에 있어서,

상기 이중 GPU들은 복수의 노드에 포함되고,

각 노드는 동일한 성능의 GPU들을 포함하고, 각 노드에 포함된 GPU들은 다른 노드에 포함된 GPU들과는 다른 성능을 가지는, 분산 학습 방법.

#### 청구항 13

제12항에 있어서,

상기 동기화하는 단계는,

동일 GPU 내에서 상기 복수의 가상 스트림 작업자 중 메인 가상 스트림 작업자를 통해 복수의 가상 스트림 작업자 별로 구한 상기 신경망 모델의 로컬 그래디언트를 합산하여 집적된 그래디언트(aggregated gradient)를 구하는 단계;

전체 축소 연산(All-reduce operation)을 통해, 상기 동일 GPU 내에서 구한 집적된 그래디언트, 상기 GPU와 동일한 노드에 속한 동종 GPU들에서 구한 집적된 그래디언트 및 상기 GPU와 상이한 노드에 속한 이중 GPU들에서 구한 집적된 그래디언트들을 평균하여 글로벌 그래디언트를 구하는 단계; 및

상기 글로벌 그래디언트에 기초하여 상기 이중 GPU들 각각에 부여된 상기 신경망 모델의 파라미터를 업데이트함으로써 상기 신경망 모델의 파라미터를 동기화하는 단계를 포함하는, 분산 학습 방법.

#### 청구항 14

제10항에 있어서,

상기 로컬 그래디언트를 연산하는 단계는,

GA(Gradient Accumulation) 기법을 이용하는 경우, GA 스텝수( $n_{GA}$ )( $n_{GA}$ : 2 이상의 자연수임)만큼의 로컬 그래디언트를 누적하고, 누적된 로컬 그래디언트의 평균을 상기 로컬 그래디언트로 구하는 단계를 포함하는, 분산 학습 방법.

#### 청구항 15

제10항에 있어서,

상기 동기화하는 단계는,

상기 이중 GPU들 각각에서, 기 설정된 방식을 통해 복수의 가상 스트림 작업자들 중 메인 가상 스트림 작업자를 이용하여 나머지 가상 스트림 작업자들로 업데이트된 상기 신경망 모델의 파라미터를 복사하는 단계를 포함하며,

상기 기 설정된 방식은,

상기 신경망 모델의 모든 레이어에 대한 전체 축소 연산(All-reduce operation)이 완료된 이후에 모든 레이어에 대한 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 순차 방식(sequential method); 및

상기 신경망 모델의 일부 레이어에 대한 전체 축소 연산이 완료되면 전체 축소 연산이 완료된 레이어에 대한 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 중첩 방식(overlapping method) 중 어느 하나를 포

함하는, 분산 학습 방법.

#### 청구항 16

제14항에 있어서,

주어진 로컬 배치 사이즈에 따라 상기 복수의 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 상기 GA 스텝수( $n_{ga}$ )를 설정하는 단계를 더 포함하는, 분산 학습 방법.

#### 청구항 17

제16항에 있어서,

상기 복수의 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 상기 GA 스텝수( $n_{ga}$ )를 설정하는 단계는,

기 정해진 최대값 이하에서 성능이 최대가 되는 상기 복수의 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 상기 GA 스텝수( $n_{ga}$ )를 구하는 단계를 포함하는, 분산 학습 방법.

#### 청구항 18

제16항에 있어서,

상기 복수의 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 상기 GA 스텝수( $n_{ga}$ )를 설정하는 단계는,

상기 이중 GPU들 각각에 대하여, 다양한 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 다양한 GA 스텝수( $n_{ga}$ )의 조합으로 구성된 검색 공간을 구성하는 단계; 및

상기 검색 공간 내에서 다이내믹 프로그래밍 기법을 이용하여 성능이 최대가 되는 상기 복수의 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 상기 GA 스텝수( $n_{ga}$ )를 구하는 단계를 포함하는, 분산 학습 방법.

### 발명의 설명

#### 기술 분야

[0001] 본 출원은, 이중 GPU들 기반의 분산 학습 장치 및 방법에 관한 것이다.

#### 배경 기술

[0002] 분산 학습이란 학습 속도를 높이기 위해 여러 GPU들(Graphic Processing Units)에 나누어 학습하는 것을 말한다. 이러한 분산 학습 방식에는 DP(Data Parallelism) 기반의 분산 학습 방법과 MP(Model Parallelism) 기반의 분산 학습 방법이 존재하며, DP 기반의 분산 학습 방법은 학습 데이터를 여러 GPU에 나누어 학습하는 방식을 의미하며, MP 기반의 분산 학습 방법은 모델을 여러 GPU에 나누어 학습하는 방식을 의미한다.

[0003] 이들 중 DP 기반의 분산 학습 방법의 경우 학습 시스템의 성능을 높이기 위해서 이중 GPU들의 성능(예: 연산력 및 메모리 용량)에 따라 서로 다른 로컬 배치 사이즈의 데이터를 사용하는 기법이 제안되었다. 하지만, 이 기술은 데이터 처리량(throughput)을 향상시킬 수 있으나, 수렴율(convergence rate)에 영향을 주어 목표 정확도(target accuracy)에 도달하지 못하거나, 도달하는데 훨씬 더 많은 에폭(epoch)을 실행해야 하는 문제가 있다.

[0004] 또한, 수렴율에 영향이 큰 배치 사이즈(batch size)를 사용자가 원하는 대로 설정할 수 있도록, 이중 GPU와 작업자(worker)의 개념을 분리하는 기법들이 제안되었으나, 이 경우 데이터 처리량이 하락할 수 있는 문제가 있다.

### 선행기술문헌

#### 특허문헌

[0005] (특허문헌 0001) 공개특허 제10-2023-0069010호("정규화를 시행하는 딥 신경망 모델 학습 장치 및 그 방법", 공개일: 2023년05월18일)

## 발명의 내용

### 해결하려는 과제

[0006] 본 발명의 일 실시 형태에 의하면, 기존 DP 기반의 분산 학습 방식에 비해 데이터 처리량을 높이고, 수렴율을 보존 또는 높일 수 있음과 동시에 데이터 처리량을 최대화할 수 있는 이중 GPU들 기반의 분산 학습 장치 및 방법을 제공한다.

### 과제의 해결 수단

[0007] 본 발명의 일 실시 형태에 의하면, 이중 GPU들 기반의 분산 학습 장치에 있어서, 상기 이중 GPU들 각각은, 매 반복(iteration)마다 멀티 스트림에서 실행되는 복수의 가상 스트림 작업자(Virtual Stream Worker, VSW) - 상기 복수의 가상 스트림 작업자 각각은 상기 멀티 스트림 중 서로 다른 스트림에서 동시에 실행됨 - 별로 신경망 모델(Neural Network Model)의 로컬 그래디언트를 연산하고, 복수의 가상 스트림 작업자 별로 구한 상기 신경망 모델의 로컬 그래디언트에 기초하여 복수의 가상 스트림 작업자에 대해 상기 신경망 모델의 파라미터를 동기화하며, 상기 복수의 가상 스트림 작업자의 수는, 성능에 따라 상기 이중 GPU 별로 상이한 값을 가지는, 분산 학습 장치가 제공된다.

[0008] 본 발명의 일 실시 형태에 의하면, 이중 GPU들 기반의 분산 학습 방법에 있어서, 상기 이중 GPU들 각각에서, 매 반복(iteration)마다 멀티 스트림에서 실행되는 복수의 가상 스트림 작업자(Virtual Stream Worker, VSW) - 상기 복수의 가상 스트림 작업자 각각은 상기 멀티 스트림 중 서로 다른 스트림에서 동시에 실행됨 - 별로 신경망 모델(Neural Network Model)의 로컬 그래디언트를 연산하는 단계; 및 상기 이중 GPU들 각각에서, 복수의 가상 스트림 작업자 별로 구한 상기 신경망 모델의 로컬 그래디언트에 기초하여 복수의 가상 스트림 작업자에 대해 상기 신경망 모델의 파라미터를 동기화하는 단계를 포함하며, 상기 복수의 가상 스트림 작업자의 수는, 성능에 따라 상기 이중 GPU 별로 상이한 값을 가지는, 분산 학습 방법이 제공된다.

[0009] 본 발명의 일 실시 형태에 의하면, 상기 분산 학습 방법을 컴퓨터 상에서 수행하기 위한 컴퓨터 프로그램을 저장한 컴퓨터로 판독 가능 기록 매체가 제공된다.

### 발명의 효과

[0010] 본 발명의 일 실시 형태에 의하면, DP 기반의 분산 학습을 수행할 때, 복수의 가상 스트림 작업자 각각은 멀티 스트림 중 서로 다른 스트림에서 동시에 실행시키되, 복수의 가상 스트림 작업자의 수는 성능에 따라 이중 GPU 별로 상이한 값을 가지도록 함으로써, 데이터 처리량을 높이고, 수렴율을 보존 또는 높일 수 있다.

[0011] 또한, 본 발명의 일 실시 형태에 의하면, 동일한 크기의 로컬 배치 사이즈의 데이터를 처리하면서도 성능(데이터 처리량)을 최대화할 수 있다.

### 도면의 간단한 설명

[0012] 도 1은 본 발명의 일 실시 형태에 따른 분산 학습의 개념을 설명하기 위한 도면이다.

도 2는 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 장치의 구성도이다.

도 3a는 본 발명의 일 실시 형태에 따라 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 방식 중 순차 방식을 설명하기 위한 도면이다.

도 3b는 본 발명의 일 실시 형태에 따라 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 방식 중 중첩 방식을 설명하기 위한 도면이다.

도 4는 본 발명의 일 실시 형태에 따라 복수의 가상 스트림 작업자의 수 및 GA 스텝수를 구하기 위한 의사 코드를 도시하고 있다.

도 5는 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 방법을 설명하는 흐름도이다.

도 6은 도 5의 단계 S520을 구체화한 도면이다.

도 7은 본 발명의 일 실시 형태에 따라 복수의 가상 스트림 작업자의 수 및 GA 스텝수를 구하는 과정을 설명하

는 흐름도이다.

도 8은 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 장치를 전체적으로 또는 부분적으로 구현할 수 있는 컴퓨팅 장치의 블록도이다.

### 발명을 실시하기 위한 구체적인 내용

- [0013] 이하, 도면을 참조하여 본 발명의 구체적인 실시형태를 설명하기로 한다. 이하의 상세한 설명은 본 명세서에서 기술된 방법, 장치 및/또는 시스템에 대한 포괄적인 이해를 돕기 위해 제공된다. 그러나 이는 예시에 불과하며 본 발명은 이에 제한되지 않는다.
- [0014] 본 발명의 실시예들을 설명함에 있어서, 본 발명과 관련된 공지기술에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략하기로 한다. 그리고 후술하는 용어들은 본 발명에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다. 상세한 설명에서 사용되는 용어는 단지 본 발명의 실시예들을 기술하기 위한 것이며, 결코 제한적이어서는 안 된다. 명확하게 달리 사용되지 않는 한, 단수 형태의 표현은 복수 형태의 의미를 포함한다. 본 설명에서, "포함" 또는 "구비"와 같은 표현은 어떤 특성들, 숫자들, 단계들, 동작들, 요소들, 이들의 일부 또는 조합을 가리키기 위한 것이며, 기술된 것 이외에 하나 또는 그 이상의 다른 특성, 숫자, 단계, 동작, 요소, 이들의 일부 또는 조합의 존재 또는 가능성을 배제하도록 해석되어서는 안 된다.
- [0015] 도 1은 본 발명의 일 실시 형태에 따른 분산 학습의 개념을 설명하기 위한 도면이다.
- [0016] 도 1을 참조하면, 분산 학습 장치(100)는 기계 학습(machine learning)을 통해 신경망 모델(20)을 학습시키는 것에 의해 학습된 신경망 모델(30)을 생성할 수 있다. 신경망 모델(20)에 대한 학습은 준비된 데이터(예: 학습 데이터)를 이용하여 신경망 모델(20)을 학습함으로써 신경망 모델의 손실이 가장 최소가 되는 지점을 찾아 가는 최적화의 과정이다.
- [0017] 신경망 모델(20)은 학습되기 이전의 신경망 모델(또는 학습되지 않은 신경망 모델)로서, 각 레이어들의 연산 및 파라미터(예: 연결 가중치) 등이 확정되지 않은 신경망 모델이다. 신경망 모델(20)은 복수의 신경망 모델 레이어(또는 간단히 '레이어')들을 포함할 수 있다. 신경망 모델(20)은 예를 들어 심층 신경망 모델(deep neural network, DNN), 컨볼루션 신경망 모델(convolutional neural network, CNN), 재귀적 신경망 모델(recurrent neural network, RNN), RBM(restricted boltzmann machine), DBN(deep belief network), BRDNN(bidirectional recurrent deep neural network), 심층 Q-네트워크(deep Q-networks) 또는 이들 중 둘 이상의 조합 중 하나일 수 있으나, 전술한 예에 한정되지 않는다. 신경망 모델(20)은 하드웨어 구조 및/또는 소프트웨어 구조를 포함할 수 있다.
- [0018] 분산 학습 장치(100)는 데이터베이스(10)에 저장된 학습 데이터(training data)를 이용하여 신경망 모델(20)에 대해 기계 학습을 수행할 수 있다. 한 번의 학습 과정(즉, 한 번의 반복(iteration))마다 신경망 모델(20)에 배치(batch) 사이즈의 학습 데이터가 입력되고, 신경망 모델(20)은 입력된 학습 데이터에 기초하여 계산된 결과 데이터를 출력한다. 분산 학습 장치(100)는 신경망 모델(20)의 결과 데이터에 기초하여 손실 함수(loss function)의 값을 최소화하는 방향으로 확률적 그라디언트 하강법(stochastic gradient descent)에 따라 학습을 수행할 수 있다. 분산 학습 장치(100)는 신경망 모델(20)로부터 출력된 결과 데이터와 검증 데이터 간의 비교 결과에 기초하여 신경망 모델 손실(neural network loss)을 결정하고, 신경망 모델 손실을 최소화하는 방향으로 신경망 모델(20)의 파라미터들(예: 가중치(weight))을 업데이트할 수 있다. 분산 학습 장치(100)는 이러한 과정을 각각의 학습 데이터에 대해 반복하여 수행함으로써 학습된 신경망 모델(30)을 생성할 수 있다.
- [0019] 일 실시예에서, 분산 학습 장치(100)는 오차 역전파 기법(error backpropagation)에 따라 신경망 모델(20)을 학습시킬 수 있다. 오차 역전파 기법은 신경망 모델(20)을 구성하는 레이어들에 대해 포워드(forward) 방향으로 계산한 결과 값과 목적하는 값간의 차이에 기초하여 계산된 오차(또는 손실)를 해당 레이어들에 대해 백워드(backward) 방향으로 거슬러 가며 오차를 최소화하는 방향으로 신경망 모델(20)의 파라미터를 조정하는 기법이다. 여기서, 포워드 방향은 신경망 모델(20)의 입력 레이어로부터 출력 레이어로 향하는 방향을 나타내고, 백워드 방향은 출력 레이어로부터 입력 레이어로 향하는 방향을 나타낸다.
- [0020] 분산 학습 장치(100)는 학습을 수행할 때 복수의 GPU를 이용하여 DP(Data Parallelism) 기반의 분산 학습을 수행할 수 있다. 이러한 DP 기반의 분산 학습을 통해 신경망 모델의 학습을 보다 빠르게 수행할 수 있다.

- [0021] 이하 도 2를 참조하여 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 장치(100)를 더욱 상세하게 설명한다.
- [0022] 도 2는 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 장치(100)의 구성도이다.
- [0023] 분산 학습 장치(100)는 프로세서(예를 들면, 컴퓨터, 마이크로 프로세서, CPU, ASIC, 로직 회로 등) 및 프로세서에 의해 실행될 때 상술한 제어부(112)의 기능들을 제공하는 소프트웨어 명령어들을 저장한 메모리를 포함할 수 있다. 여기서, 프로세서와 메모리는 별개의 반도체 회로로 구현되거나, 프로세서와 메모리는 하나의 집적된 반도체 회로로 구현될 수 있다. 프로세서는 하나 이상일 수 있다.
- [0024] 도 2에 도시된 바와 같이, 이중 GPU들 기반의 분산 학습 장치(100)는 이중 GPU(Graphic Processing Unit) 클러스터 환경(120)에서 동작할 수 있다.
- [0025] 이러한 이중 GPU 클러스터 환경(120)에 의하면, 분산 학습 장치(100)는 복수의 노드(Node 1, Node 2, ..., Node N)를 포함하며, 각 노드는 동일한 성능의 GPU들을 포함하고, 각 노드에 포함된 GPU들은 다른 노드에 포함된 GPU들과는 다른 성능을 가질 수 있다. 예를 들면, 도 2에서, 노드 1에 포함된 GPU 집합(220)의 GPU들(A1, A2, A3, A4)은 모두 동일한 성능을 가진다. 또한, 노드 2에 포함된 GPU 집합의 GPU들(B1, B2, B3, B4)은 모두 동일한 성능을 가지며, 노드 1에 포함된 GPU들(A1, A2, A3, A4)과는 다른 성능을 가질 수 있다. 여기서, 성능이란 GPU의 연산력 및 메모리 용량 중 적어도 하나를 의미할 수 있다.
- [0026] 또한, 이중 GPU들 각각은 매 반복(iteration) 마다 멀티 스트림에서 실행되는 복수의 가상 스트림 작업자(Virtual Stream Worker, VSW) 별로 신경망 모델(Neural Network Model)의 로컬 그라디언트를 연산할 수 있다.
- [0027] 예를 들면, 도 2에서, GPU A2(300)의 경우 복수의 가상 스트림 작업자(VSW1 내지 VSW N, 310, 320, 330) 각각의 연산부(311, 321, 331)는 신경망 모델의 레이어들에 대해 포워드 방향(예: 입력 레이어에서 출력 레이어로의 방향, Fi)의 연산을 수행할 수 있다.
- [0028] 이후, 복수의 가상 스트림 작업자(VSW1 내지 VSW N, 310, 320, 330) 각각의 연산부(311, 321, 331)는 포워드 방향의 연산의 결과에 기초하여 신경망 모델의 손실을 구할 수 있다. 입력 데이터를 처리한 신경망 모델의 결과 데이터와 목적하는 검증 데이터간의 차이에 기초하여 신경망 모델의 손실이 결정될 수 있다.
- [0029] 이후 복수의 가상 스트림 작업자(VSW1 내지 VSW N, 310, 320, 330) 각각의 연산부(311, 321, 331)는 신경망 모델의 손실에 기초하여 신경망 모델의 레이어들에 대해 백워드 방향(예: 출력 레이어에서 입력 레이어로의 방향, Bi)의 연산을 수행하는 것에 의해 신경망 모델의 각 레이어별로 로컬 그라디언트(local gradient)를 결정할 수 있다. 로컬 그라디언트는 하나의 GPU(예: A2)에서 구한 그라디언트를 의미한다.
- [0030] 여기서, 복수의 가상 스트림 작업자 각각은 매 반복(iteration) 마다 동일한 크기의 로컬 배치 사이즈( $B_L$ )의 데이터를 처리할 수 있다. 또한, 복수의 가상 스트림 작업자 각각은 멀티 스트림 중 서로 다른 스트림에서 동시에 실행될 수 있으며, 복수의 가상 스트림 작업자의 수는 GPU의 성능에 따라 GPU 별로 상이한 값을 가질 수 있다. 예를 들면, 도 2에서, 노드 1에 포함된 GPU A2(300)의 복수의 가상 스트림 작업자의 수는 VSW1 부터 VSW N까지 N개일 수 있으며, 노드 2에 포함된 GPU B2(400)의 복수의 가상 스트림 작업자의 수는 VSW1 부터 VSW M까지 M개일 수 있다. 일 예로, GPU의 성능이 높을 수록 복수의 가상 스트림 작업자의 수는 그에 비례하여 많아질 수 있다.
- [0031] 이러한 멀티 스트림을 구현하기 위해 노드 내에는 CPU(Central Processing Unit)가 더 포함될 수 있으며, CPU 내에서 다중 쓰레드 프로세스(예: process 2)에 의해 복수의 디스패처(dispatcher<sub>1</sub> 내지 dispatcher<sub>N</sub>)가 실행되며, 복수의 디스패처(dispatcher<sub>1</sub> 내지 dispatcher<sub>N</sub>) 각각은 멀티 스트림 중 서로 다른 스트림에서 복수의 가상 스트림 작업자 각각을 실행시킬 수 있다. 예를 들면, 도 2에서, 노드 1에 포함된 CPU(210)의 다중 쓰레드 프로세서(process<sub>2</sub>, 211)에 의해 복수의 디스패처(dispatcher<sub>1</sub> 내지 dispatcher<sub>N</sub>, 211a 내지 211c)가 실행될 수 있으며, 복수의 디스패처(dispatcher<sub>1</sub> 내지 dispatcher<sub>N</sub>, 211a 내지 211c) 각각은 멀티 스트림 중 서로 다른 스트림에서 복수의 가상 스트림 작업자(VSW1 내지 VSW N, 310, 320, 330)를 각각 실행시킬 수 있다.
- [0032] 한편, GA(Gradient Accculation) 기법을 이용하는 경우에는 GA 스텝수( $n_{GA}$ )( $n_{GA}$ : 2 이상의 자연수임)만큼의 로컬 그라디언트를 누적하고, 누적된 로컬 그라디언트의 평균을 로컬 그라디언트로 구할 수 있다.
- [0033] 예를 들면, 도 2에서, GPU A2(300)의 경우 포워드 방향(Fi)의 연산 및 백워드 방향(Bi)의 연산은 1번만 이루어

지므로, GA 기법을 이용하지 않는 경우이다. 반면, GPU B2(400)의 경우 포워드 방향(Fi)의 연산 및 백워드 방향(Bi)의 연산이 2번 이루어지므로 GA 스텝수( $n_{GA}$ )는 2일 수 있다. 이 경우 각각의 Fi-Bi, Fi-Bi를 통해 구한 2개의 로컬 그라디언트를 누적하고, 누적된 로컬 그라디언트의 평균을 로컬 그라디언트로 구할 수 있다.

- [0034] 이후, 동기화부(312)는 신경망 모델의 로컬 그라디언트에 기초하여 신경망 모델의 파라미터를 동기화할 수 있다. 상술한 동기화부(312)는 복수의 가상 스트림 작업자(VSW1 내지 VSW N, 310, 320, 330) 중 메인 가상 스트림 작업자(310)에 포함될 수 있다.
- [0035] 예를 들면, 도 2에서, GPU A2(300)의 경우 메인 가상 스트림 작업자(VSW1, 310)의 동기화부(312)는 복수의 가상 스트림 작업자(VSW1 내지 VSW N, 310, 320, 330) 별로 구한 신경망 모델의 로컬 그라디언트를 합산하여 집적된 그라디언트(aggregated gradient)를 구할 수 있다(local gradient aggregate, 312a).
- [0036] 이후, 동기화부(312)는 전체 축소 연산(All-reduce operation, 312b)을 통해, 동일 GPU 내에서 구한 집적된 그라디언트, GPU와 동일한 노드에 속한 동종 GPU들에서 구한 집적된 그라디언트(AR1) 및 GPU와 상이한 노드에 속한 이종 GPU들에서 구한 집적된 그라디언트(AR2)을 평균하여 글로벌 그라디언트를 구할 수 있다.
- [0037] 이후, 동기화부(312)는 글로벌 그라디언트에 기초하여 다른 가상 스트림 작업자(VSW2, VSWN) 각각의 신경망 모델의 파라미터를 업데이트함으로써 신경망 모델의 파라미터(예: 가중치)를 동기화할 수 있다(Weight Update, S312c). 이와 같이 글로벌 그라디언트에 기초하여 신경망 모델의 파라미터를 업데이트하기 위한 함수로는 옵티마이저(optimizer)가 사용될 수 있다.
- [0038] 본 발명의 일 실시 형태에 의하면, 동기화부(312)는 기 설정된 방식을 통해 복수의 가상 스트림 작업자들 중 메인 가상 스트림 작업자를 이용하여 나머지 가상 스트림 작업자들로 업데이트된 신경망 모델의 파라미터를 복사할 수 있다.
- [0039] 여기서, 기 설정된 방식은 신경망 모델의 모든 레이어에 대한 전체 축소 연산(All-reduce operation)이 완료된 이후에 모든 레이어에 대한 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 순차 방식(sequential method) 및 신경망 모델의 일부 레이어에 대한 전체 축소 연산이 완료되면 전체 축소 연산이 완료된 레이어에 대한 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 중첩 방식(overlapping method) 중 어느 하나를 포함할 수 있다.
- [0040] 도 3a는 본 발명의 일 실시 형태에 따라 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 방식 중 순차 방식을 설명하기 위한 도면이며, 도 3b는 본 발명의 일 실시 형태에 따라 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 방식 중 중첩 방식을 설명하기 위한 도면이다. 편의상 도 3a 및 도 3b에서는 2개의 가상 스트림 작업자(VSW1, VSW2)만이 도시되어 있으며, 이들 중 VSW1은 메인 가상 스트림 작업자일 수 있다.
- [0041] 도 3a 내지 도 3b에서, 도면부호 351, 361은 VSW1, VSW2에서의 레이어별 백워드 방향의 연산을, 도면부호 352, 362는 VSW1, VSW2에서의 레이어별 포워드 방향의 연산을, 도면부호 353, 363은 누적(sum)을 통한 집적된 레이어별 로컬 그라디언트를, 도면부호 354, 364는 메인 가상 스트림 작업자(SW1)에서 전체 축소 연산을 통해 구한 레이어별 글로벌 그라디언트를, 도면부호 355, 365는 메인 가상 스트림 작업자(SW1)에서 글로벌 그라디언트에 기초하여 구한 업데이트된 파라미터(예: 가중치)를, 도면부호 356, 366은 메인 가상 스트림 작업자(SW1)에서 다른 가상 스트림 작업자(VSW2)로 업데이트된 파라미터를 복사하는 것을 의미한다.
- [0042] 도 3a에 도시된 순차 방식(sequential method)에 의하면, 메인 가상 스트림 작업자(VSW1)는 신경망 모델의 모든 레이어에 대한 전체 축소 연산이 완료된 이후에 모든 레이어에 대한 업데이트된 파라미터(355)를 다른 가상 스트림 작업자들로 순차로 복사(copy, 356)할 수 있다.
- [0043] 또는, 도 3b에 도시된 중첩 방식(overlapping method)에 의하면, 메인 가상 스트림 작업자(VSW1)는 신경망 모델의 일부 레이어에 대한 전체 축소 연산이 완료되면 전체 축소 연산이 완료된 일부 레이어에 대한 업데이트된 파라미터(365 중 U4, U5)를 다른 가상 스트림 작업자(VSW2)로 복사(copy, 366)할 수 있다.
- [0044] 상술한 순차 방식 또는 중첩 방식은 신경망 모델의 레이어 특성에 따라 선택적으로 사용될 수 있다.
- [0045] 한편, 본 발명의 일 실시 형태에 의하면, 분산 학습 장치(100)는 설정부(110)를 더 포함할 수 있다.
- [0046] 상술한 설정부(110)는 신경망 모델(DNN) 및 하이퍼 파라미터(Hybrid Parameter, HP)를 입력받고, 주어진 로컬 배치 사이즈( $B_L$ )에 따라 복수의 가상 스트림 작업자의 수( $n_{VSW}$ ) 및 GA 스텝수( $n_{GA}$ )를 설정한 후 이를 각 노드

(Node 1, Node 2, Node N)로 전달할 수 있다. 하이퍼 파라미터는 신경망 모델을 학습시키기 위한 파라미터이며, 예를 들면 최소 로컬 배치 사이즈, 최대 로컬 배치 사이즈, 학습률 스케일링 규칙(learning rate scaling rule) 등이 포함될 수 있다.

[0047] 이후, 설정부(110)는 기 정해진 글로벌 배치 사이즈의 최대값(maxB<sub>G</sub>) 이하에서 성능이 최대가 되는 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>) 및 GA 스텝수(n<sub>GA</sub>)를 구할 수 있다. 기 정해진 글로벌 배치 사이즈의 최대값(maxB<sub>G</sub>)은 상술한 학습률 스케일링 규칙에 의해 미리 정해진 값일 수 있다.

[0048] 구체적으로, 설정부(110)는 이중 GPU들 각각에 대하여, 주어진 로컬 배치 사이즈(B<sub>L</sub>)에 따라 다양한 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>) 및 다양한 GA 스텝수(n<sub>GA</sub>)의 조합으로 구성된 검색 공간(S)을 구성하고, 다이내믹 프로그래밍 기법을 이용하여 검색 공간(S) 내에서 성능이 최대가 되는 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>) 및 GA 스텝수(n<sub>GA</sub>)를 구할 수 있다.

[0049] 도 4에는 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>) 및 GA 스텝수(n<sub>GA</sub>)를 구하기 위한 의사 코드(pseudo code)가 예시적으로 도시되어 있다.

[0050] 도 4의 주요 부분을 설명하면, 설정부(110)는 사용자로부터 신경망 모델(D), GPU 집합(Galloc) 및 최소 및 최대 로컬 배치 사이즈(minB<sub>L</sub>, maxB<sub>L</sub>)가 주어지면, 최소 및 최대 로컬 배치 사이즈(minB<sub>L</sub>, maxB<sub>L</sub>)의 범위 내에서 다양한 로컬 배치 사이즈(B<sub>L</sub>)에 대한 성능(예: 데이터 처리량)을 예측함으로써 PU들 각각에 대한 최적의 구성 C\*을 구할 수 있다.

[0051] 구체적으로, 설정부(110)는 로컬 배치 사이즈(B<sub>L</sub>)가 주어지면, CONFIGURE(Galloc, D, B<sub>L</sub>) 함수를 호출한다(라인 1).

[0052] 라인 2 내지 라인 4에서, 변수 S는 GPU들 각각의 성능 타입, 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>), GA 스텝수(n<sub>GA</sub>), 연산 시간(t) 및 가상 스트림 작업자의 수(n<sub>w</sub>)를 포함하는 복수의 작업자 및 스텝수의 조합(s)들을 구비한 검색 공간일 수 있다. 연산 시간(t)은 GA 스텝수(n<sub>GA</sub>) \* 로컬 배치 사이즈(B<sub>L</sub>)의 데이터의 단위 연산 시간이며, 단위 연산 시간이란 포워드 방향(Fi)-백워드 방향(Bi)의 연산을 1번 수행하는데 걸리는 시간을 의미할 수 있다. 그리고, 가상 스트림 작업자의 수(n<sub>w</sub>)는 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>) \* GA 스텝수(n<sub>GA</sub>)를 의미할 수 있다.

[0053] 한편, 변수 C는 GPU별 s들의 집합이며, C.n<sub>w</sub>는 C의 멤버 중 n<sub>w</sub>를 의미하며, C.t는 GPU들 중 연산 시간(t)이 가장 긴 GPU의 연산 시간을 의미한다.

[0054] 이후, 설정부(110)는 각 GPU g에 대하여 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>)를 최대값(Maxn<sub>vs<sub>w</sub></sub>(g, D, BL)) 이내에서 1씩 증가시키고, 부가하여 GA 스텝수(n<sub>GA</sub>)를 최대값(MAXn<sub>GA</sub>)까지 1씩 증가시키면서 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>) 및 GA 스텝수(n<sub>GA</sub>)의 다양한 조합(s)으로 구성된 검색 공간(S)을 구성한다(라인 9 내지 라인 16).

[0055] 한편, 라인 10의 로그 회귀 분석(logarithmic regression)은 하기 수학식 1과 같을 수 있다.

[0056] [수학식 1]

$$T_{pred} = (B_L * n_{vs\subscript{w}}) / (a * \ln(n_{vs\subscript{w}}) + b)$$

[0058] 여기서, T<sub>pred</sub>는 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>)에 따른 예측된 단위 연산 시간을, B<sub>L</sub>은 로컬 배치 사이즈를, n<sub>vs<sub>w</sub></sub>는 가상 스트림 작업자의 수를, a, b는 상수를 의미한다. 본 발명에서 로그 회귀 분석을 이용하는 이유는 하나의 GPU에서 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>)가 증가할 경우 성능(예: 데이터 처리량)은 일정 지점에서 더 이상 향상되지 않기 때문이다.

[0059] 이후, 설정부(110)는 다이내믹 프로그래밍 기법을 이용하여 하기 수학식 2에 따라 검색 공간(S) 내에서 다양한 s를 함수 F에 대입함으로써, 성능이 최대가 되는 가상 스트림 작업자의 수(n<sub>vs<sub>w</sub></sub>) 및 GA 스텝수(n<sub>GA</sub>)의 조합으로 구성된 GPU들의 조합인 구성 C를 구할 수 있다.

[수학식 2]

$$F(s, C) = \max \begin{cases} (B_L * C.n_w) / C.t \\ (B_L * (s.n_w + (C - C').n_w) / \max(s.t, (C - C').t) \end{cases}$$

수학식 2에서, 첫번째 수식의  $(B_L * C.n_w) / C.t$ 는 이전 단계에서 구한 값이다. 분자는 글로벌 배치 사이즈를 의미하고, 분모는 다양한 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 다양한 GA 스텝수( $n_{GA}$ )의 조합으로 구성된 GPU들의 조합인 구성 C 중 가장 긴 연산 시간을 가지는 GPU의 연산 시간을 의미한다.

두번째 수식의  $(B_L * (s.n_w + (C - C').n_w) / \max(s.t, (C - C').t)$ 는 현 단계에서 입력된 s에 대하여 구한 값이다. 분자인  $(B_L * (s.n_w))$ 는 현 단계에서 입력된 s에 대하여 구한 글로벌 배치 사이즈를,  $B_L * (C - C').n_w$ 는 현 단계에서 입력된 s를 가진 GPU는 배제하고 나머지 GPU들에 대하여 구한 글로벌 배치 사이즈를 의미한다.

즉, 첫번째 수식과 두번째 수식을 비교한 후 더 큰 값을 첫번째 수식에 대한 값으로 대체는 방식으로 가상 스트림 작업자의 수( $n_{vsw}$ ) 및 GA 스텝수( $n_{GA}$ )의 조합으로 구성된 GPU들의 최적 조합인 구성 C를 구할 수 있다.

상술한 바와 같이, 본 발명의 일 실시 형태에 의하면, DP 기반의 분산 학습을 수행할 때, 복수의 가상 스트림 작업자 각각은 멀티 스트림 중 서로 다른 스트림에서 동시에 실행시키되, 복수의 가상 스트림 작업자의 수는 성능에 따라 이중 GPU 별로 상이한 값을 가지도록 함으로써, 데이터 처리량을 높이고, 수렴율을 보존 또는 높일 수 있다.

본 발명의 일 실시 형태에 의하면, 동일한 크기의 로컬 배치 사이즈의 데이터를 처리하면서도 성능(데이터 처리량)을 최대화할 수 있다.

한편, 도 5는 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 방법을 설명하는 흐름도이다. 도 6은 도 5의 단계 S520을 구체화한 도면이며, 도 7은 본 발명의 일 실시 형태에 따라 복수의 가상 스트림 작업자의 수 및 GA 스텝수를 구하는 과정을 설명하는 흐름도이다.

이하, 도 1 내지 도 7를 참조하여 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 방법(S500)을 설명한다. 다만, 발명의 간명화를 위해 도 1 내지 도 4와 관련하여 중복된 설명은 생략한다.

도 1 내지 도 7를 참조하면, 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 방법(S500)은 이중 GPU들 각각에서, 매 반복 마다 멀티 스트림에서 실행되는 복수의 가상 스트림 작업자 별로 신경망 모델의 로컬 그라디언트를 연산하는 단계에 의해 시작될 수 있다(S510). 여기서, 복수의 가상 스트림 작업자 각각은 멀티 스트림 중 서로 다른 스트림에서 동시에 실행될 수 있다.

또한, 복수의 가상 스트림 작업자의 수는 성능에 따라 이중 GPU 별로 상이한 값을 가질 수 있으며, 복수의 가상 스트림 작업자는 동일한 크기의 로컬 배치 사이즈( $B_L$ )의 데이터를 처리할 수 있음은 상술한 바와 같다.

또한, 이중 GPU들은 복수의 노드들에 포함되고, 각 노드는 동일한 성능의 GPU들을 포함하고, 각 노드에 포함된 GPU들은 다른 노드에 포함된 GPU들과는 다른 성능을 가질 수 있음은 상술한 바와 같다.

한편, 상술한 로컬 그라디언트를 연산하는 단계(S510)는 GA 기법을 이용하는 경우 GA 스텝수만큼의 로컬 그라디언트를 누적하고, 누적된 로컬 그라디언트의 평균을 로컬 그라디언트로 구할 수 있음은 상술한 바와 같다.

이후, 이중 GPU들 각각에서, 복수의 가상 스트림 작업자 별로 구한 신경망 모델의 로컬 그라디언트에 기초하여 복수의 가상 스트림 작업자에 대해 신경망 모델의 파라미터를 동기화할 수 있다(S520).

이하, 도 6을 참조하여 단계 S520을 설명한다.

구체적으로, 동일 GPU 내에서 복수의 가상 스트림 작업자 중 메인 가상 스트림 작업자를 통해 복수의 가상 스트림 작업자 별로 구한 신경망 모델의 로컬 그라디언트를 합산하여 집적된 그라디언트를 구할 수 있다(S521).

이후, 전체 축소 연산을 통해, 동일 GPU 내에서 구한 집적된 그라디언트, GPU와 동일한 노드에 속한 동종 GPU들에서 구한 집적된 그라디언트 및 GPU와 상이한 노드에 속한 이중 GPU들에서 구한 집적된 그라디언트들을 평균하

여 글로벌 그라디언트를 구할 수 있다(S522).

- [0077] 이후, 글로벌 그라디언트에 기초하여 이중 GPU들 각각에 부여된 신경망 모델의 파라미터를 업데이트함으로써 신경망 모델의 파라미터를 동기화할 수 있다(S523).
- [0078] 한편, 상술한 단계 S520은 이중 GPU들 각각에서, 기 설정된 방식을 통해 복수의 가상 스트림 작업자들 중 메인 가상 스트림 작업자를 이용하여 나머지 가상 스트림 작업자들로 업데이트된 신경망 모델의 파라미터를 복사하는 단계를 더 포함할 수 있다.
- [0079] 여기서, 기 설정된 방식은, 신경망 모델의 모든 레이어에 대한 전체 축소 연산이 완료된 이후에 모든 레이어에 대한 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 순차 방식(sequential method) 및 신경망 모델의 일부 레이어에 대한 전체 축소 연산이 완료되면 전체 축소 연산이 완료된 레이어에 대한 업데이트된 파라미터를 다른 가상 스트림 작업자들로 복사하는 중첩 방식(overlapping method) 중 어느 하나를 포함할 수 있음은 상술한 바와 같다.
- [0080] 본 발명의 일 실시 형태에 의하면, 상술한 분산 학습 방법(S500)은 주어진 로컬 배치 사이즈에 따라 복수의 가상 스트림 작업자의 수 및 GA 스텝수를 설정하는 단계를 더 포함할 수 있다.
- [0081] 이때, 복수의 가상 스트림 작업자의 수 및 GA 스텝수를 설정하는 단계는 기 정해진 최대값 이하에서 성능이 최대가 되는 복수의 가상 스트림 작업자의 수 및 GA 스텝수를 구하는 단계를 포함할 수 있다.
- [0082] 구체적으로, 도 7에 도시된 바와 같이, 복수의 가상 스트림 작업자의 수 및 상기 GA 스텝수를 설정하는 단계는, 이중 GPU들 각각에 대하여, 다양한 가상 스트림 작업자의 수 및 다양한 GA 스텝수의 조합으로 구성된 검색 공간을 구성한다(S710). 이후 검색 공간 내에서 다이나믹 프로그래밍 기법을 이용하여 성능이 최대가 되는 복수의 가상 스트림 작업자의 수 및 GA 스텝수를 구할 수 있다(S720).
- [0083] 상술한 바와 같이, 본 발명의 일 실시 형태에 의하면, DP 기반의 분산 학습을 수행할 때, 복수의 가상 스트림 작업자 각각은 멀티 스트림 중 서로 다른 스트림에서 동시에 실행시키되, 복수의 가상 스트림 작업자의 수는 성능에 따라 이중 GPU 별로 상이한 값을 가지도록 함으로써, 데이터 처리량을 높이고, 수렴율을 보존 또는 높일 수 있다.
- [0084] 본 발명의 일 실시 형태에 의하면, 동일한 크기의 로컬 배치 사이즈의 데이터를 처리하면서도 성능(데이터 처리량)을 최대화할 수 있다.
- [0085] 도 8은 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 장치를 전체적으로 또는 부분적으로 구현할 수 있는 컴퓨팅 장치(800)의 블록도이다.
- [0086] 도 8에 도시된 바와 같이, 컴퓨팅 장치(800)는 적어도 하나의 프로세서(801), 컴퓨터 판독 가능 저장 매체(802) 및 통신 버스(803)를 포함한다.
- [0087] 프로세서(801)는 컴퓨팅 장치(800)로 하여금 앞서 언급된 예시적인 실시예에 따라 동작하도록 할 수 있다. 예컨대, 프로세서(801)는 컴퓨터 판독 가능 저장 매체(802)에 저장된 하나 이상의 프로그램들을 실행할 수 있다. 상기 하나 이상의 프로그램들은 하나 이상의 컴퓨터 실행 가능 명령어를 포함할 수 있으며, 상기 컴퓨터 실행 가능 명령어는 프로세서(801)에 의해 실행되는 경우 컴퓨팅 장치(800)로 하여금 예시적인 실시예에 따른 동작들을 수행하도록 구성될 수 있다.
- [0088] 컴퓨터 판독 가능 저장 매체(802)는 컴퓨터 실행 가능 명령어 내지 프로그램 코드, 프로그램 데이터 및/또는 다른 적합한 형태의 정보를 저장하도록 구성된다. 컴퓨터 판독 가능 저장 매체(802)에 저장된 프로그램(802a)은 프로세서(801)에 의해 실행 가능한 명령어의 집합을 포함한다. 일 실시예에서, 컴퓨터 판독 가능 저장 매체(802)는 메모리(랜덤 액세스 메모리와 같은 휘발성 메모리, 비휘발성 메모리, 또는 이들의 적절한 조합), 하나 이상의 자기 디스크 저장 디바이스들, 광학 디스크 저장 디바이스들, 플래시 메모리 디바이스들, 그 밖에 컴퓨팅 장치(800)에 의해 액세스되고 원하는 정보를 저장할 수 있는 다른 형태의 저장 매체, 또는 이들의 적합한 조합일 수 있다.
- [0089] 통신 버스(803)는 프로세서(801), 컴퓨터 판독 가능 저장 매체(802)를 포함하여 컴퓨팅 장치(800)의 다른 다양한 컴포넌트들을 상호 연결한다.
- [0090] 컴퓨팅 장치(800)는 또한 하나 이상의 입출력 장치(804)를 위한 인터페이스를 제공하는 하나 이상의 입출력 인터페이스(805) 및 하나 이상의 네트워크 통신 인터페이스(806)를 포함할 수 있다. 입출력 인터페이스(805) 및

네트워크 통신 인터페이스(806)는 통신 버스(803)에 연결된다. 네트워크는 셀룰러 네트워크, 예컨대 GSM(Global System for Mobile Communications), EDGE(Enhanced Data Rates for GSM Evolution), GPRS(General Packet Radio Service), CDMA(Code Division Multiple Access), 시분할(Time Division)-CDMA(TD-CDMA), UMTS(Universal Mobile Telecommunications System), LTE(Long Term Evolution), 또는 다른 셀룰러 네트워크 중 어느 하나일 수 있다.

[0091] 입출력 장치(804)는 입출력 인터페이스(805)를 통해 컴퓨팅 장치(800)의 다른 컴포넌트들에 연결될 수 있다. 예시적인 입출력 장치(804)는 포인팅 장치(마우스 또는 트랙패드 등), 키보드, 터치 입력 장치(터치패드 또는 터치스크린 등), 음성 또는 소리 입력 장치, 다양한 종류의 센서 장치 및/또는 촬영 장치와 같은 입력 장치, 및/또는 디스플레이 장치, 프린터, 스피커 및/또는 네트워크 카드와 같은 출력 장치를 포함할 수 있다. 예시적인 입출력 장치(804)는 컴퓨팅 장치(800)를 구성하는 컴포넌트로서 컴퓨팅 장치(800)의 내부에 포함될 수도 있고, 컴퓨팅 장치(800)와는 구별되는 별개의 장치로 컴퓨팅 장치(800)와 연결될 수도 있다.

[0092] 한편, 본 발명의 실시예는 본 명세서에서 기술한 방법들을 컴퓨터상에서 수행하기 위한 프로그램, 및 상기 프로그램을 포함하는 컴퓨터 판독 가능 기록매체를 포함할 수 있다. 상기 컴퓨터 판독 가능 기록매체는 프로그램 명령, 로컬 데이터 파일, 로컬 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체는 본 발명을 위하여 특별히 설계되고 구성된 것들이거나, 또는 컴퓨터 소프트웨어 분야에서 통상적으로 사용 가능한 것일 수 있다. 컴퓨터 판독 가능 기록매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM, DVD와 같은 광 기록 매체, 및 롬, 램, 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 상기 프로그램의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함할 수 있다.

[0093] 이상에서 본 발명의 대표적인 실시예들을 상세하게 설명하였으나, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 상술한 실시예에 대하여 본 발명의 범주에서 벗어나지 않는 한도 내에서 다양한 변형이 가능함을 이해할 것이다. 그러므로 본 발명의 권리범위는 설명된 실시예에 국한되어 정해져서는 안 되며, 후술하는 특허 청구범위뿐만 아니라 이 특허청구범위와 균등한 것들에 의해 정해져야 한다.

## 부호의 설명

[0094] 10: 데이터베이스  
20: 학습전 신경망 모델  
30: 학습된 신경망 모델  
100: 이중 GPU들 기반의 분산 학습 장치  
110: 설정부  
120: 이중 클러스터 환경  
200: 노드 1  
210: CPU  
211: 프로세스 1  
211a 내지 211c: 디스패처  
300: GPU A<sub>2</sub>  
310, 320, 330: 가상 스트림 작업자  
312: 동기화부  
312a: local gradient aggregate  
312b: All-reduce  
312c: Weight Update  
311, 321, 331: 연산부

400: GPU B<sub>2</sub>

800: 본 발명의 일 실시 형태에 따른 이중 GPU들 기반의 분산 학습 장치를 전체적으로 또는 부분적으로 구현할 수 있는 컴퓨팅 장치의 블록도

801: 프로세서

802: 컴퓨터 판독 가능 저장 매체

802a: 프로그램

803: 통신 버스

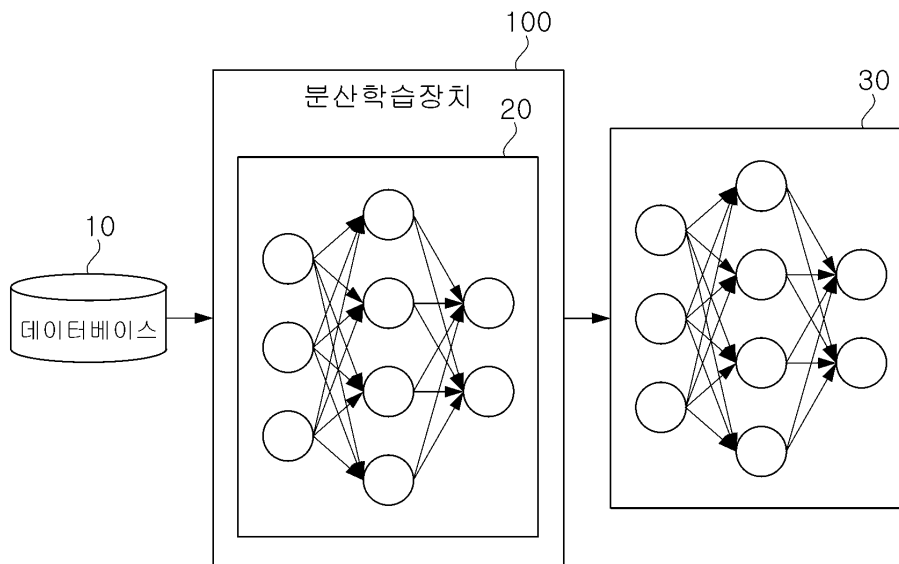
804: 입출력 장치

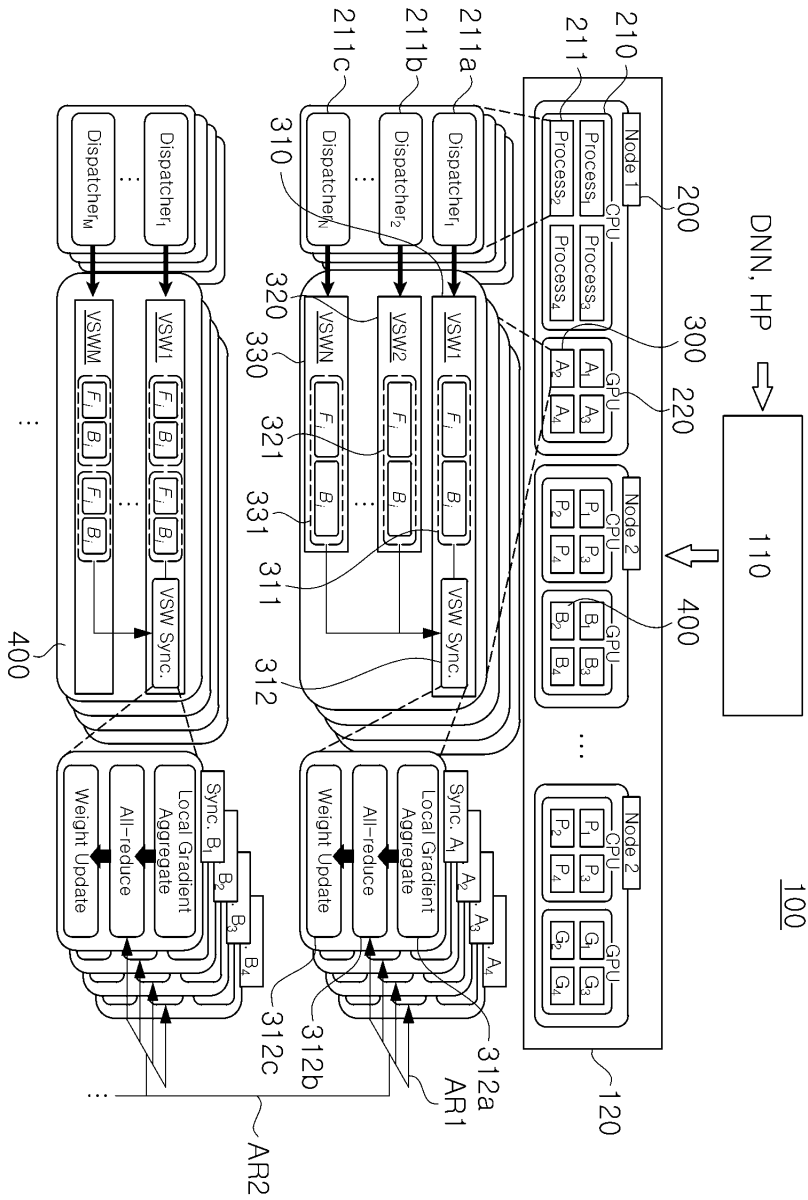
805: 입출력 인터페이스

806: 네트워크 통신 인터페이스

**도면**

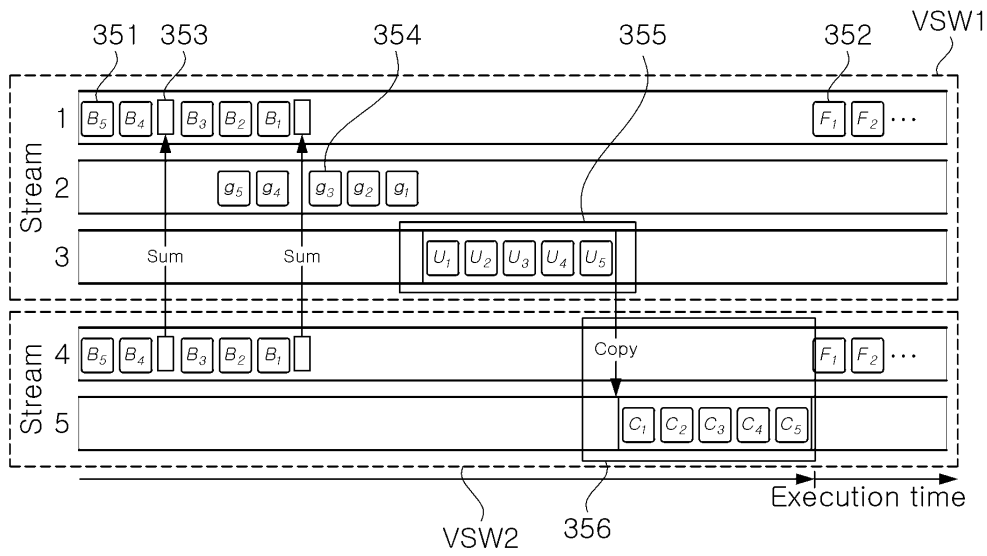
**도면1**



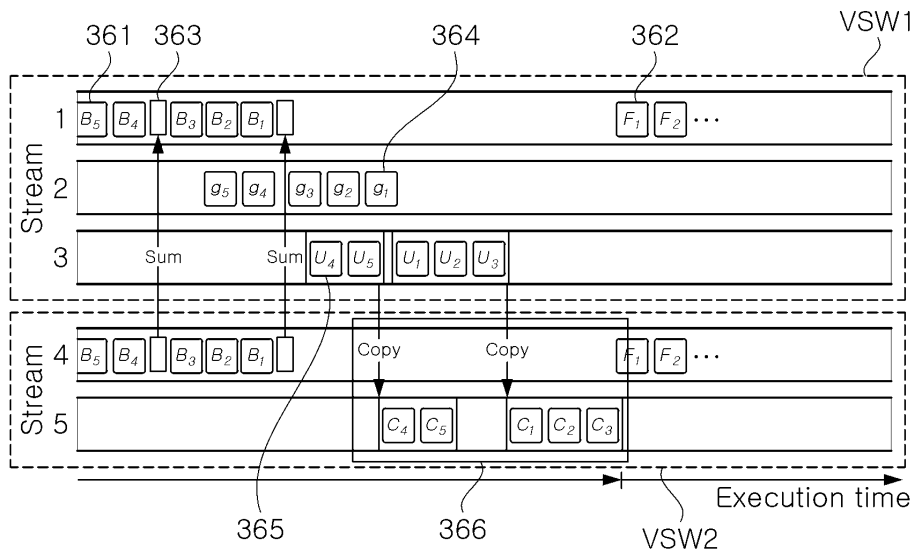


도면2

도면3a



도면3b



도면4

---

### Algorithm 1 VSW Configurator

---

**Input:**  $G_{alloc} = \{G_0, G_1, \dots, G_{N_{gpu}-1}\}$ , DNN model  $D$ , Local batch size  $B_L$ ;  
**Output:** Optimal configuration  $C^*$ , Global batch size  $B_G^*$

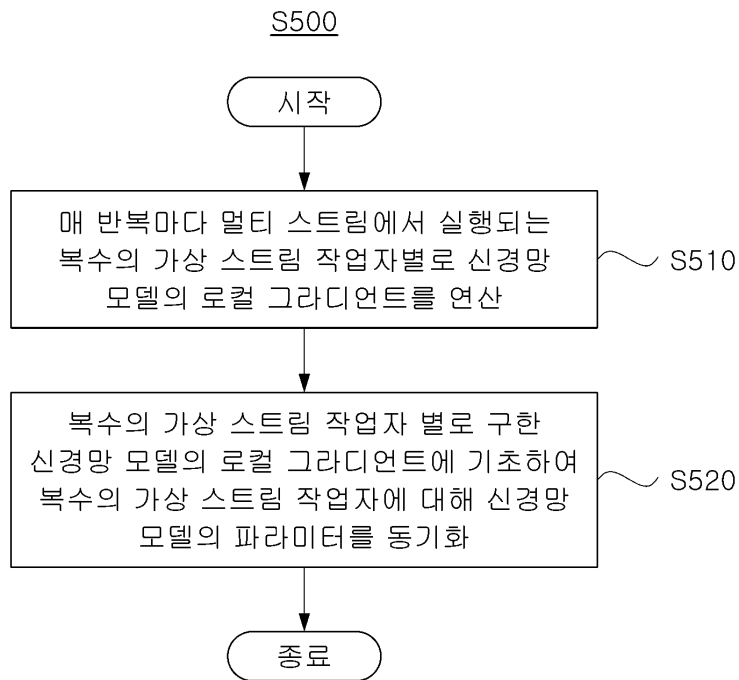
```

1: function CONFIGURE( $G_{alloc}, D, B_L$ )
2:   var:  $S = \{s | s = (type(G_i), n_{vsw}, n_{GA}, t, n_w), G_i \in G_{alloc}\}$ 
3:    $C = \{s_i | i \in 0, 1, \dots, N_{gpu} - 1\}$ 
4:    $C.n_w = \sum s_i.n_w, C.t = \max_{0 \leq i \leq N_{gpu}-1} (s_i.t)$ 
5:   for  $g$  in  $G_{alloc}$  do
6:     if  $MAXn_{vsw}(g, D, B_L) == 0$  then
7:       Continue
8:     end if
9:     for  $n_{vsw} = 1$  to  $MAXn_{vsw}(g, D, B_L)$  do
10:       $T_{pred} := \text{PREDICT}(n_{vsw});$  ▷ Logarithmic regression
11:      for  $n_{GA} = 1$  to  $MAXn_{GA}$  do
12:         $t = T_{pred} * n_{GA}$ 
13:         $S.add((g, n_{vsw}, n_{GA}, t, n_{vsw} * n_{GA}))$ 
14:      end for
15:    end for
16:  end for
17:  for  $s$  in Pruning( $S$ ) do
18:    Compute  $F(s, C)$ 
19:  end for
20:   $C^* = \text{argmax } F, B_G^* = B_L * C.n_w$ 
21:  Return  $C^*, B_G^*$ 
22: end function

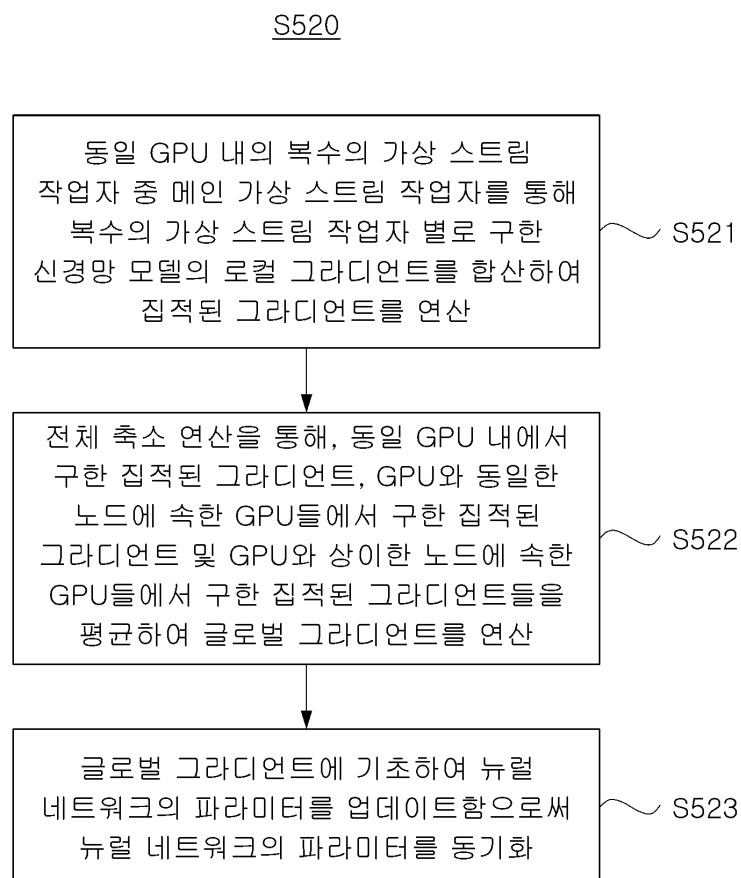
```

---

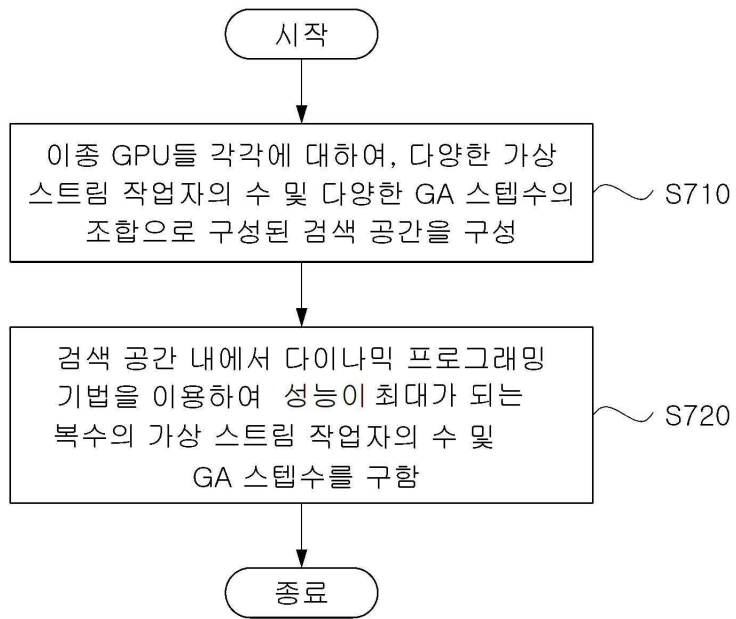
도면5



도면6



도면7



도면8

