(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0021104
(43) 공개일자 2020년02월27일

(51) 국제특허분류(Int. Cl.)
G06F 9/50 (2018.01) *G06N 20/00* (2019.01)
G06N 3/04 (2006.01) *G06N 3/063* (2006.01)
G06N 3/08 (2006.01)
(52) CPC특허분류
G06F 9/5066 (2013.01)
G06F 9/5038 (2013.01)
(21) 출원번호 10-2020-7004981(분할)
(22) 출원일자(국제) 2016년10월28일
심사청구일자 없음
(62) 원출원 특허 10-2018-7015068
원출원일자(국제) 2016년10월28일
심사청구일자 2018년05월28일
(85) 번역문제출일자 2020년02월20일
(86) 국제출원번호 PCT/US2016/059334
(87) 국제공개번호 WO 2017/075360
국제공개일자 2017년05월04일
(30) 우선권주장
62/247,703 2015년10월28일 미국(US)
62/253,046 2015년11월09일 미국(US)

(71) 출원인
구글 엘엘씨
미국 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600 (우:94043)
(72) 발명자
바함 폴 로날드
미국 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600 (우:94043)
바수데반 비제이
미국 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600 (우:94043)
(74) 대리인
박장원

전체 청구항 수 : 총 1 항

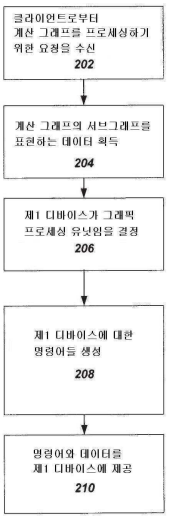
(54) 발명의 명칭 **계산 그래프들의 스트림-기반 가속기 프로세싱**

(57) 요 약

계산 그래프 시스템에 의해, 계산 그래프를 프로세싱하기 위한 요청을 수신하는 단계; 상기 계산 그래프의 서브그래프를 표현하는 데이터를 획득하는 단계, 상기 계산 그래프는 복수의 노드(node)들과 방향 에지(directed edge)들을 포함하며, 각 노드는 각각의 동작을 표현하며, 각 방향 에지는 각각의 제1 노드를 각각의 제2 노드에

(뒷면에 계속)

대 표 도 - 도2



연결하며, 계산 그래프 시스템에서 상기 서브그래프는 플레이서에 의해 제1 디바이스에 할당되며; 상기 제1 디바이스가 복수의 스트림들을 가지는 하드웨어 가속기를 포함한다는 것을 결정하는 단계; 상기 결정에 응답하여, 명령어들을 생성하는 단계를 포함하며, 상기 명령어들은 상기 제1 디바이스에 의해 실행될 때 상기 제1 디바이스로 하여금: 상기 서브그래프에서 각 노드에 의해 표현된 상기 동작을 각각의 스트림에 할당하게 하고; 그리고 상기 할당에 따라 상기 서브그래프에서 각 노드에 의해 표현된 상기 동작들을 수행하게 하는 컴퓨터 저장 매체에 인코딩된 컴퓨터 프로그램을 포함하는 방법, 시스템, 장치.

(52) CPC특허분류
G06N 20/00 (2019.01)
G06N 3/0454 (2013.01)
G06N 3/063 (2013.01)
G06N 3/084 (2013.01)
H05K 999/99 (2013.01)

명 세 서

청구범위

청구항 1

방법으로서,

계산 그래프 시스템에 의해, 계산 그래프를 프로세싱하기 위한 요청을 수신하는 단계;

상기 계산 그래프의 서브그래프를 표현하는 데이터를 획득하는 단계, 상기 계산 그래프는 복수의 노드(node)들과 방향 에지(directed edge)들을 포함하며, 각 방향 에지는 각각의 제1 노드를 각각의 제2 노드에 연결하며, 상기 각각의 제2 노드는 상기 각각의 제1 노드에 표현된 동작의 출력을 입력으로서 수신하는 동작을 표현하며, 상기 서브그래프는 상기 계산 그래프 시스템에서 플레이서에 의해 제1 디바이스에 할당되며;

상기 제1 디바이스가 복수의 스트림들을 가지는 하드웨어 가속기를 포함한다는 것을 결정하는 단계;

상기 제1 디바이스가 복수의 스트림들을 가지는 하드웨어 가속기를 포함한다는 것을 결정함에 응답하여, 명령어들을 생성하는 단계, 상기 명령어들은 상기 제1 디바이스에 의해 실행될 때 상기 제1 디바이스로 하여금:

상기 서브그래프에서 각 노드에 의해 표현된 상기 동자글 상기 하드웨어 가속기의 상기 복수의 스트림들 중 각각의 스트림에 할당하게 하고; 그리고

상기 할당에 따라 상기 서브그래프에서 상기 노드들에 의해 표현된 상기 동작들을 수행하게 하며; 및

상기 명령어들 및 상기 데이터를 상기 제1 디바이스에 제공하는 단계를 포함하는 것을 특징으로 하는 방법.

발명의 설명

기 술 분 야

배 경 기 술

[0001] 본 명세서는 서브그래프를 다수의 스트림들을 가지는 가속기 디바이스 예를 들면, 그래픽 프로세싱 유닛(GPU)에 할당함으로써 신경 네트워크를 나타내는 계산 그래프를 프로세싱하는 것 및/또는 상기 프로세싱된 계산 그래프를 모델 입력을 프로세싱하기 위해 사용하는 것과 관련된다.

[0002] 신경 네트워크들은 모델들의 하나 이상의 레이어들을 이용하여 수신된 입력에 대한 출력 예를 들면, 하나 이상의 분류들을 생성하는 기계 학습 모델들이다. 일부 신경 네트워크들은 출력 레이어에 더하여 하나 이상의 히든 레이어들을 포함한다. 각 히든 레이어의 출력은 네트워크에서 다음 레이어 즉, 다음 히든 레이어 또는 네트워크의 출력 레이어에 대한 입력으로서 사용된다. 네트워크의 각 레이어는 상기 레이어에 대한 각각의 세트의 현재 값들에 따라 수신된 입력으로부터 출력을 생성한다.

[0003] 존재하는 시스템들에서, 계산 그래프들의 동작들은 개별적 디바이스에 의해 프로세싱될 수 있다. 일부 구현예들에서, 디바이스는 GPU이다. 디바이스는 동작들 예를 들면, 입력들로부터 레이어에서 출력들을 생성하는 것을 수행하고 상기 동작들로부터의 출력들을 메모리에 저장하는 프로세서를 가질 수 있다. 많은 수와 크기의 동작들이 계산 그래프에서 출력들을 생성하기 위해 일반적으로 요구되기 때문에, 하나의 디바이스가 그래프의 동작들을 프로세싱하기 위해 상당량의 시간을 가질 수 있다.

발명의 내용

[0004] 일반적으로, 본 명세서는 스트림-기반 가속기 디바이스 예를 들면, GPU를 사용하여 계산 그래프의 서브그래프들을 프로세싱하는 시스템 또는 방법을 기술한다.

[0005] 일반적으로, 본 명세서에서 기술된 주제의 일 혁신적 양태는 방법으로 이용될 수 있으며, 상기 방법은 계산 그래프를 프로세싱하기 위한 요청을 수신하는 단계; 상기 계산 그래프의 서브그래프를 표현하는 데이터를 획득하

는 단계, 상기 계산 그래프는 복수의 노드(node)들과 방향 에지(directed edge)들을 포함하며, 각 노드는 각각의 동작을 표현하며, 각 방향 에지는 상기 각각의 제1 노드에 의해 표현된 동작의 출력을 입력으로서 수신하는 동작을 표현하는 각각의 제1 노드를 각각의 제2 노드에 연결하며, 계산 그래프 시스템에서 상기 서브그래프는 플레이서에 의해 제1 디바이스에 할당되며; 상기 제1 디바이스가 복수의 스트림들을 가지는 하드웨어 가속기를 포함한다는 것을 결정하는 단계; 상기 제1 디바이스가 복수의 스트림들을 가지는 하드웨어 가속기를 포함한다는 것을 결정함에 응답하여, 명령어들을 생성하는 단계, 상기 명령어들은 상기 제1 디바이스에 의해 실행될 때 상기 제1 디바이스로 하여금: 상기 서브그래프에서 각 노드에 의해 표현된 상기 동작을 상기 그래픽 프로세싱 유닛의 복수의 스트림들에서의 각각의 스트림에 할당하게 하고; 그리고 상기 할당에 따라 상기 서브그래프에서 각 노드에 의해 표현된 상기 동작들을 수행하게 하며; 및 상기 명령어들 및 상기 데이터를 상기 제1 디자이스에 제공하는 단계의 액션들을 포함한다. 본 양태의 방법은 컴퓨터로 구현되는 방법이다. 본 양태의 방법은 하나 이상의 컴퓨팅 디바이스들, 예를 들면 계산 그래프 시스템을 포함하는 하나 이상의 컴퓨팅 디바이스들에 의해 수행될 수 있다.

[0006] 구현예들은 다음 구성들 중 하나 이상을 포함할 수 있다. 요청은 상기 서브그래프에서 하나 이상의 각각의 노드들로부터 하나 이상의 특정한 출력들을 식별하는 것을 특정하며, 제1 디바이스로부터, 하나 이상의 특정한 출력들을 수신하는 단계; 및 상기 하나 이상의 특정한 출력들을 클라이언트에 제공하는 단계를 더 포함한다. 명령어들은 제1 디바이스로 하여금 상기 하나 이상의 특정한 출력들을 상기 제1 디바이스의 메모리에 저장하게 한다. 상기 서브그래프에 대한 동작들은 신경 네트워크에 대한 부분적 추론 또는 계산들 트레이닝을 포함한다. 서브그래프의 노드들의 그룹을 식별하기 위해 체인 구조에서 상기 서브그래프를 분석하는 단계를 더 포함하며, 명령어들은 제1 디바이스로 하여금 상기 노드들의 그룹을 하나의 스트림에 할당하게 한다. 상기 할당은: 상기 서브그래프에서 복수의 방향 에지들을 가지는 제1 노드를 출력으로서 식별하기 위해 상기 서브그래프를 분석하는 것을 포함하며; 상기 명령어들은 상기 제1 디바이스로 하여금 상기 방향 에지들 각각에 대해, 상기 방향 에지가 그래픽 프로세싱 유닛의 고유 스트림(unique stream)을 포인팅하는 노드에 할당하게 한다. 상기 명령어들은 상기 제1 디바이스로 하여금 각 노드에 대해, 상기 노드에 대한 상기 방향 에지들에 기초하여, 상기 노드에 의해 표현된 상기 동작에 의해 소비되는 상기 그래픽 프로세싱 유닛의 메모리 리소스들의 각각의 양을 결정하게 하며, 상기 할당은 상기 메모리 리소스들의 상기 각각의 양에 적어도 기초한다. 상기 명령어들은 상기 제1 디바이스로 하여금 노드에 의해 표현된 특정한 동작이 특정한 스트림에서 종료되었다고 결정하게 하며; 상기 특정한 동작이 종료되었다고 결정함에 응답하여: 자유화될 상기 특정한 동작에 의해 소비된 메모리의 제1 양을 결정하게 하며; 비할당 노드들의 그룹 각각에 대해, 상기 비할당 노드에 의해 소비된 메모리의 각각의 예측된 양을 결정하게 하며; 상기 비할당 노드들의 그룹으로부터, 상기 메모리의 제1양의 사용을 최대화하는 상기 메모리의 예측된 양을 가지는 제1 비할당 노드를 결정하게 하며; 그리고 상기 제1 비할당 노드에 의해 표현된 동작을 상기 특정한 스트림에 할당하게 한다.

[0007] 구현예에서 상기 방법은: 모델 입력을 수신하는 단계; 및 상기 하드웨어 가속기에 의해, 상기 서브그래프의 상기 노드들에 의해 표현된 동작들에 따라 상기 모델 입력을 프로세싱하는 단계를 더 포함한다.

[0008] 다른 양태에서, 본 명세서에 기술된 본 발명은 상기 제1 양태의 방법에 의해 획득된 프로세싱된 계산 그래프에 대응하는 기계 학습 모델들을 제공하고, 상기 기계 학습 모델을 사용하여 모델 입력을 프로세싱하는 액션들을 포함할 수 있는 방법들에 수록될 수 있다.

[0009] 다른 양태에서, 본 명세서에 기술된 본 발명은 하드웨어 가속기에 의해 상기 제1 양태의 방법에 의해 획득된 프로세싱된 계산 그래프의 서브그래프를 실행하는 액션들을 포함할 수 있는 방법들에 수록될 수 있다.

[0010] 이들 양태들에서, 계산 그래프는 기계 학습 모델 예를 들면, 신경 네트워크의 표현일 수 있다.

[0011] 다른 혁신적 양태는 계산 그래프의 서브그래프를 표현하는 데이터를 수신하는 단계, 상기 계산 그래프는 복수의 노드(node)들과 방향 에지(directed edge)들을 포함하며, 각 노드는 각각의 동작을 표현하며, 각 방향 에지는 각각의 제1 노드를 각각의 제2 노드에 연결하며, 이는 상기 각각의 제1 노드에 의해 표현된 동작의 출력을 입력으로서 수신하는 동작을 표현하며, 상기 서브그래프는 계산 그래프 시스템에서 플레이서(placer)에 의해 그래픽 프로세싱 유닛에 할당되며; 상기 서브그래프에서 각 노드에 의해 표현된 상기 동작을 상기 그래픽 프로세싱 유닛의 상기 복수의 스트림들에서의 각각의 스트림에 할당하는 단계; 및 상기 할당에 따라 상기 서브그래프에서 상기 노드들에 의해 표현된 상기 동작들을 수행하는 단계의 액션들을 포함한다.

[0012] 구현예들은 다음 구성들 중 하나 이상을 포함할 수 있다. 상기 서브그래프에서 하나 이상의 각각의 노드들로부터 하나 이상의 특정한 출력들을 식별하는 요청을 수신하는 단계; 및 상기 하나 이상의 특정한 출력들을 클라이

언트에 제공하는 단계. 체인 구조에서 상기 서브그래프의 노드들의 그룹을 식별하는 데이터를 수신하는 단계; 및 상기 노드들의 그룹을 하나의 스트림에 할당하는 단계. 상기 할당은: 상기 서브그래프에서 복수의 방향 에지들을 가지는 제1 노드를 출력들로서 식별하는 데이터를 수신하는 것; 및 상기 방향 에지들 각각에 대해, 상기 방향 에지가 상기 그래픽 프로세싱 유닛의 고유 스트림(unique stream)을 포인팅하는 노드에 할당하는 것을 포함. 각 노드에 대해, 상기 노드에 대한 상기 방향 에지들에 기초하여, 상기 노드에 의해 표현된 상기 동작에 의해 소비되는 상기 그래픽 프로세싱 유닛의 메모리 리소스들의 각각의 양을 결정하는 단계, 상기 할당은 상기 메모리 리소스들의 상기 각각의 양에 적어도 기초함. 노드에 의해 표현된 특정한 동작이 특정한 스트림에서 종료되었다고 결정하는 단계; 상기 특정한 동작이 종료되었다고 결정함에 응답하여, 자유화될 상기 특정한 동작에 의해 소비된 메모리의 제1 양을 결정하는 단계; 비할당 노드들의 그룹 각각에 대해, 상기 비할당 노드에 의해 소비된 메모리의 각각의 예측된 양을 결정하는 단계; 상기 비할당 노드들의 그룹으로부터, 상기 메모리의 제1양의 사용을 최대화하는 상기 메모리의 예측된 양을 가지는 제1 비할당 노드를 결정하는 단계; 및 상기 제1 비할당 노드에 의해 표현된 동작을 상기 특정한 스트림에 할당하는 단계.

[0013] 이들 및 다른 양태들의 다른 구현예들은 방법들의 액션들을 수행하도록 구성된 대응하는 시스템, 장치 및 컴퓨터 저장 디바이스들(비일시적 저장 디바이스이거나 일시적 저장 디바이스일 수 있는)에 인코딩된 컴퓨터 프로그램을 포함한다.

[0014] 본 명세서에 기술된 본 발명의 특정한 실시예들은 다음의 이점들을 실현하도록 구현될 수 있다. 신경 네트워크의 동작들 예를 들면, 입력으로부터 추론을 생성하는 동작은 노드들 및 방향 에지들의 계산 그래프로서 표현될 수 있다. 시스템은 동작들을 효율적으로 수행하기 위해 이 계산 그래프 표현을 프로세싱한다. 계산 그래프가 다수의 스트림들을 가지기 때문에 시스템은 이 효율성을 달성한다. 다수의 스트림들을 사용하는 것은 논리적으로 독립적인 동작들이 재정렬되거나 또는 동시적으로 실행되게 한다. 시스템이 전체 계산을 위해 종단 간 지연을 저하시키는 목표를 가지는 경우, 예시적 시스템은 논리적으로 독립적인 동작들을 재정렬할 수 있다. 시스템이 보다 많은 처리량을 달성하기 위한 목표를 가지는 경우, 예시적 시스템은 동작들을 동시적으로 실행할 수 있다. 계산 그래프는 병렬 동작들에 대해 통상적은 표현보다 더 쉽게 파티션화될 수 있다. 도시로서, 계산 그래프의 서브그래프들은 고유 디바이스들에 할당될 수 있고, 그것들 각각은 각각의 서브그래프에서의 동작들을 수행하여, 신경 네트워크의 동작들을 수행하는데 요구되는 전체 시간을 감소시킨다.

[0015] 서브그래프가 할당되는 디바이스는 GPU일 수 있다. 서브그래프는 상기 서브그래프의 동작들을 보다 효율적으로 수행하기 위해 GPU의 다수의 스트림들에 파티션화될 수 있다. 본 명세서의 발명의 하나 이상의 실시예들의 세부사항은 첨부 도면과 아래의 설명에서 기술된다. 본 발명의 다른 구성들, 양태들 및 이점들은 설명, 도면 및 청구항으로부터 명백해질 것이다. 양태들 및 구현예들은 조합될 수 있고, 하나의 양태 또는 구현예의 맥락에서 기술된 구성들은 다른 양태들 또는 구현예들의 맥락에서 구현될 수 있다는 것이 인식될 것이다.

도면의 간단한 설명

[0016] 도 1은 계산 그래프들로서 표현된 신경 네트워크들에 대한 동작들을 분배하는 예시적 계산 그래프 시스템을 도시한다.

도 2는 GPU를 사용하여 계산 그래프의 서브그래프를 프로세싱하기 위한 예시적 프로세스의 흐름도이다.

도 3은 GPU에 의해 프로세싱되는 계산 그래프의 예시적 서브그래프를 도시한다.

도 4는 노드들을 스트림들에 할당하기 위한 예시적 프로세스의 흐름도이다.

다양한 도면들에서 동일한 참조 번호 및 기호는 동일한 구성요소를 표시한다.

발명을 실시하기 위한 구체적인 내용

[0017] 본 명세서는 일반적으로 분산된 방식으로 계산 그래프에 의해 표현된 동작들을 수행하는 계산 그래프 시스템을 기술한다.

[0018] 계산 그래프는 방향 에지(directed edge)들에 의해 연결된 노드들을 포함한다. 계산 그래프에서 각 노드는 동작을 표현한다. 노드로 들어오는 에지는 상기 노드로의 입력의 흐름 즉, 노드에 의해 표현된 동작에 대한 입력을 표현한다. 노드로부터 나가는 에지는 다른 노드에 의해 표현된 동작에 대한 입력으로서 사용될 노드에 의해 표현된 동작의 출력의 흐름을 표현한다. 따라서, 그래프에서 제1 노드를 그래프에서 제2 노드에 연결하는 방향 에지는 제1 노드에 의해 표현된 동작에 의해 생성된 출력이 제2 노드에 의해 표현된 동작에 대한 입력으로서 사용

된다는 것을 표시한다.

[0019] 일반적으로, 계산 그래프에서 방향 에지들을 따라 흐르는 입력과 출력들은 텐서(tensor)들이다. 텐서는 어레이(array)의 차원수(dimensionality)에 대응하는 특정 순서(order)를 가지는 수치값 또는 다른 값들 예를 들어, 스트링들의 다차원 어레이이다. 예를 들면, 스칼라 값은 0번째 순서 텐서이며, 수치값들의 벡터는 1번째 순서 텐서이며, 그리고 행렬(matrix)은 2번째 순서 텐서이다.

[0020] 일부 구현예들에서, 계산 그래프에서 표현되는 동작들은 신경 네트워크 동작들 또는 상이한 종류의 기계 학습 모델을 위한 동작들이다. 신경 네트워크는 수신된 입력의 출력을 예측하기 위한 비선형 유닛들의 하나 이상의 레이어들을 이용하는 기계 학습 모델이다. 일부 신경 네트워크들은 출력 레이어에 더하여 하나 이상의 히든 레이어들을 포함하는 딥 신경 네트워크들이다. 각 히든 레이어의 출력은 네트워크에서 다른 레이어 즉, 다른 히든 레이어, 출력 레이어 또는 둘 모두에 대한 입력으로서 사용된다. 네트워크의 일부 레이어들은 각각의 세트의 현재 값들에 따라 수신된 입력으로부터 출력을 생성하며, 네트워크의 다른 레이어들은 파라미터들을 가지지 않을 수 있다.

[0021] 예를 들면, 계산 그래프에 의해 표현된 동작들은 추론을 계산하기 위해 즉, 신경 네트워크의 레이어들을 통한 입력을 프로세싱하여 상기 입력에 대한 신경 네트워크 출력을 생성하이 위해, 신경 네트워크에 필요한 동작들일 수 있다. 다른 예시로서, 계산 그래프에 의해 표현된 동작들은 신경 네트워크의 파라미터들의 값들을 조절하기 위해 예를 들어, 파라미터들의 초기값들로부터 파라미터들의 트레이닝된 값들을 결정하기 위해 신경 네트워크 트레이닝 절차를 수행함으로써, 신경 네트워크를 트레이닝하기 위해 필요한 동작들일 수 있다. 일부 경우들에서, 예를 들어, 신경 네트워크의 트레이닝 동안에, 계산 그래프에 의해 표현된 동작들은 신경 네트워크의 다수의 레플리카(replica)들에 의해 수행된 동작들을 포함할 수 있다.

[0022] 도시로서, 이전 레이어로부터 입력을 수신하는 신경 네트워크 레이어는 파라미터 행렬과 입력 사이에서 행렬곱(matrix multiplication)을 수행하기 위해 파라미터 행렬을 사용할 수 있다. 일부 경우들에서, 이 행렬곱은 계산 그래프에서 다수의 노드들로서 표현될 수 있다. 예를 들면, 행렬곱은 다수의 곱셈과 추가 동작들로 나누어질 수 있고, 각 동작은 계산 그래프에서 상이한 노드에 의해 표현될 수 있다. 각 노드에 의해 표현된 동작은 방향 에지에서 후속 노드로 흐르는 각각의 출력을 생성할 수 있다. 최종 노드에 의해 표현된 동작이 행렬곱의 결과를 생성한 후에, 결과는 방향에지에서 다른 노드로 흐른다. 결과는 행렬곱을 수행하는 신경 네트워크 레이어의 출력과 동등하다.

[0023] 일부 경우들에서, 행렬곱은 그래프에서 하나의 노드로서 표현된다. 노드에 의해 표현된 동작들은 입력들로서 제1 방향 에지에서 입력 텐서와 제2 방향 에지에서 가중 텐서(weight tensor) 예를 들어, 파라미터 행렬을 수신할 수 있다. 일부 구현예들에서, 가중 텐서는 모델의 공유된 지속적 상태(persistent state)와 연관된다. 노드는 신경 네트워크 레이어의 출력과 동등한 제3 방향 에지에서 출력 텐서를 출력하기 위해 입력과 가중 텐서를 프로세싱한다(예를 들어, 입력과 가중 텐서의 행렬곱 수행).

[0024] 계산 그래프에서 노드들에 의해 표현될 수 있는 다른 신경 네트워크 동작들은 기타 수학적 동작들 예를 들어, 뺄셈, 나누기 및 기울기 계산; 어레이 동작들 예를 들어, 연결(concatenate), 결합(splice), 쪼개기(split) 또는 랭킹(rank); 및 신경 네트워크 빌딩 블록 동작 예를 들어, SoftMax, Sigmoid, ReLU(rectified linear unit) 또는 컨벌루션들을 포함한다.

[0025] 신경 네트워크를 계산 그래프로서 표현하는 것은 특히 신경 네트워크에 대한 동작들이 상이한 하드웨어 프로필을 가지는 다수의 디바이스들에 걸쳐서 분배된 경우, 신경 네트워크를 효율적으로 구현하기 위한 유연하고 세밀한 방식을 위해 제공된다.

[0026] 도 1은 계산 그래프들로서 표현된 신경 네트워크들에 대한 동작들을 분배하는 예시적 계산 그래프 시스템(100)을 도시한다. 시스템(100)은 이하에 기술된 시스템들, 컴포넌트들 및 기법들이 구현될 수 있는, 하나 이상의 위치들의 하나 이상의 컴퓨터들에서 컴퓨터 프로그램들로서 구현된 시스템의 예시이다.

[0027] 클라이언트(102)의 사용자는 신경 네트워크를 표현하는 계산 그래프에서 수행되는 액션들을 요청할 수 있다. 예를 들면, 클라이언트는 그래프를 세션 관리자로 등록하고, 데이터 입력을 그래프에 공급하거나 그래프의 출력들 중 하나 이상을 평가할 수 있다. 클라이언트(102)는 컴퓨터에서 실행되는 어플리케이션일 수 있다.

[0028] 요청의 부분으로서, 클라이언트(102)는 계산 그래프를 식별하는 데이터를 시스템(100)에 제공하고, 계산 그래프에서 수행될 액션들의 유형을 특정한다.

[0029] 예를 들면, 요청은 특정한 신경 네트워크에 대한 추론을 표현하는 계산 그래프를 식별하고, 추론이 수행되어야 할 입력을 식별할 수 있다.

[0030] 다른 예시로서, 요청은 특정한 신경 네트워크에 대한 트레이닝 절차를 표현하는 계산 그래프를 식별하고, 트레이닝이 수행되어야 할 트레이닝 데이터와 같은 입력을 식별할 수 있다. 이 예시에서, 트레이닝 절차를 표현하는 계산 그래프를 프로세싱하기 위한 요청을 수신하는 경우, 시스템(100)은 예를 들어, 통상적인 역전파 또는 다른 신경 네트워크 트레이닝 기법들을 사용하여, 계산 그래프의 하나 이상의 노드들에 대한 파라미터들에 대해 수정된 값들을 결정할 수 있다. 시스템(100)은 수정된 파라미터들을 디바이스의 메모리에 저장할 수 있고, 실행자(106)는 시스템(100)에서 수정된 가중치들의 주소들을 검색하고 저장할 수 있다. 추론, 트레이닝 또는 수정된 가중치를 요하는 다른 동작들을 위한 클라이언트(102)로부터 추가적 요청들에 따라, 시스템(100)은 주소들을 사용하여 수정된 가중치들에 엑세스할 수 있다.

[0031] 일부 경우들에서, 요청은 상기 요청에 대응하여 전송되어야 하는 응답을 특정할 수 있다. 예를 들면, 신경 네트워크 트레이닝 요청에 대해, 클라이언트(102)는 요청된 신경 네트워크 트레이닝 동작들이 완료되었다는 표시와, 선택적으로, 신경 네트워크의 파라미터들의 트레이닝된 값들 또는 상기 트레이닝된 값들이 클라이언트(102)에 의해 엑세스될 수 있는 메모리 위치의 표시를 요청할 수 있다. 다른 예시로서, 신경 네트워크 추론 요청에 대해, 클라이언트(102)는 계산 그래프의 하나 이상의 특정한 노드들로부터 추론 동작을 표현하는 출력값들을 요청할 수 있다.

[0032] 시스템(100)은 다수의 디바이스들(115-122)에 걸쳐서 계산 그래프에 의해 표현된 동작들을 파티션화함으로써, 특정한 출력을 생성하기 위한 동작들을 수행한다. 시스템(100)은 데이터 통신 네트워크(114) 예를 들어, LAN 또는 WAN을 통해 동작들을 다수의 디바이스들(116-122)로 파티션화한다. 디바이스들(116-122)은 동작들을 수행하며, 적용가능한 경우, 각각의 출력 또는 표시를 시스템(100)에 리턴하며, 이는 요청된 출력 또는 표시를 클라이언트(102)에 리턴할 수 있다.

[0033] 신경 네트워크 동작들을 수행하는 임의의 디바이스들 예를 들어, 디바이스들(116-122)은 명령어들과 데이터를 저장하기 위한 메모리 예를 들어, RAM과 저장된 명령어들을 실행하기 위한 프로세서를 포함할 수 있다. 일반적으로, 각 디바이스는 다른 디바이스들과 독립적으로 동작들을 수행하는 하드웨어 리소스이다. 예를 들면, 각 디바이스는 그것 고유의 프로세싱 유닛을 가진다. 디바이스들은 GPU(Graphical Processing Unit)들, CPU(Central Processing Unit)들 또는 다른 가속기들일 수 있다. 예시로서, 하나의 기계는 하나 이상의 디바이스들 예를 들어, 다수의 CPU들 및 GPU들을 호스팅할 수 있다.

[0034] 또한 각 디바이스는 각각의 계산 능력을 가질 수 있다. 즉, 디바이스들은 서로 다른 양의 메모리, 프로세싱 속도 또는 기타 아키텍처적 특징들을 가질 수 있다. 따라서, 일부 디바이스들은 다른 디바이스들은 할 수 없는 동작들을 수행할 수 있다. 예를 들면, 일부 동작들은 단지 특정한 디바이스들만 가지는 특정 양의 메모리를 요구하거나 또는 일부 디바이스들은 특정한 유형의 동작 예를 들어, 추론 동작들만 수행하도록 구성된다.

[0035] 시스템(100)에서 세션 관리자(104)는 클라이언트(102)로부터 요청을 수신하여 계산 그래프의 동작들이 수행되는 세션을 시작한다. 세션 관리자(104)는 계산 그래프의 동작들을 수행할 수 있는 디바이스들의 세트 예를 들어, 디바이스들(116-122)을 관리하며, 플레이서(108)에 동작들을 수행하기 위해 사용가능한 디바이스들의 세트를 제공할 수 있다.

[0036] 플레이서(108)는 계산 그래프에서 수행될 각 동작에 대해, 상기 동작을 수행하는 각각의 타겟 디바이스 예를 들어, 디바이스(116)과 일부 구현예들에서, 상기 동작을 수행하기 위한 상기 각각의 타겟 디바이스에 대한 시간을 결정한다. 플레이서(108)는 주어진 입력 데이터의 크기에 대해 각 사용가능한 디바이스에서 동작이 얼마나 오래 걸릴 것인지 알고 있음으로써 최적의 디바이스 할당을 수행한다. 플레이서(108)는 측정들 또는 예상 성능 모델들을 사용하여 프로세싱 시간의 예측치를 획득한다. 예를 들어, 다른 동작들은 이전 동작들의 출력들을 입력들로서 프로세싱하여, 다른 동작들은 완료될 계산 그래프에서 이전의 동작들을 요구하지만, 일부 동작들은 병렬로 수행될 수 있다.

[0037] 디바이스가 출력들을 생성하기 위해 플레이서(108)에 의해 할당된 동작들을 수행한 후에, 실행자(106)는 출력들을 검색할 수 있다. 실행자(106)는 요청에 대한 적절한 응답 예를 들어, 출력 또는 프로세싱이 완료되었다는 표시를 생성할 수 있다. 그 후, 실행자(106)는 상기 응답을 클라이언트(102)에 리턴할 수 있다. 비록 도 1이 하나의 실행자(106)를 도시하였지만, 하나의 구현예에서는, 디바이스마다 실행자가 있다. 이 실행자는 동작들이 실행가능하게 된 경우(즉, 동작들의 모든 입력들이 계산됨) 동작들을 디바이스에 발행한다. 또한, 이 구현예는 플

레이서(108)를 호출함으로써 다수의 디바이스들에서 실행하기 위해 그래프를 파티션화하고 필요한 실행자들을 생성하는 그래프 관리자를 가진다.

[0038] 또한, 세션 관리자(104)는 계산 그래프에서 수행될 동작들의 세트들을 실행자(106)에게 제공한다. 실행자(106)는 디바이스들(116-122)로부터 동작들의 그래프 실행과 관련된 런타임 통계를 주기적으로 검색한다. 실행자(106)는 상기 런타임 통계를 플레이서(108)에 제공하며, 이는 추가적 동작들의 배치(placement) 및 스케쥴을 재-최적화할 수 있다.

[0039] 도 2는 GPU를 사용하여 계산 그래프의 서브그래프를 프로세싱하기 위한 예시적 프로세스(200)의 흐름도이다. 편의상, 프로세스(200)는 하나 이상의 위치들에 위치된 하나 이상의 컴퓨터들의 시스템에 의해 수행되는 것으로서 기술될 것이다. 예를 들면, 적절하게 프로그래밍된 계산 그래프 시스템 예를 들어, 도 1의 계산 그래프 시스템(100)은 프로세스(200)를 수행할 수 있다.

[0040] 시스템은 계산 그래프를 프로세싱하기 위한 요청을 클라이언트로부터 수신한다(단계(202)). 예를 들면, 요청은 도 1을 참조하여 상기 기술된 바와 같이, 특정 입력에 대해 계산 그래프에 의해 표현된 신경 네트워크 추론을 수행하기 위한 요청, 트레이닝 데이터의 특정 세트에 대해 계산 그래프에 의해 표현된 신경 네트워크 트레이닝 동작들을 수행하기 위한 요청 또는 계산 그래프에 의해 표현된 다른 신경 네트워크 동작들을 수행하기 위한 요청일 수 있다.

[0041] 일부 경우들에서, 계산 그래프는 클라이언트로부터의 요청들로 보내어진다. 다른 경우들에서, 요청은 계산 그래프를 식별하고 시스템은 식별된 그래프를 표현하는 데이터를 메모리로부터 검색한다.

[0042] 시스템은 계산 그래프를 다수의 서브그래프들로 파티션화할 수 있다. 일부 구현예에서, 서브그래프들을 요청을 보내는 클라이언트에 의해 특정되며, 시스템은 상기 특정에 따라 계산 그래프를 파티션화한다. 일부 다른 구현예에서, 시스템은 계산 그래프를 파티션화하여, 각 서브그래프가 다른 서브그래프들에 비해 동작들을 수행하기 위해 유사한 양의 리소스들을 요구하도록 한다.

[0043] 시스템은 예를 들어, 도 1의 플레이서(108)를 사용하여 각 서브그래프를 사용가능한 디바이스에 할당할 수 있다.

[0044] 시스템은 파티션화된 계산 그래프로부터 계산 그래프의 특정한 서브그래프를 표현하는 데이터를 획득한다(단계(204)). 데이터는 데이터베이스 또는 시스템의 메모리로부터 획득될 수 있다. 예시로서, 특정한 서브그래프의 동작들은 부분적 추론 또는 트레이닝 계산들을 표현한다.

[0045] 시스템은 서브그래프가 할당되는 디바이스는 그래픽 프로세싱 유닛 또는 다수의 스트림들을 가지는 다른 하드웨어 가속기 디바이스임을 결정한다(단계(206)). 예시로서, 시스템은 계산 그래프에 할당될 디바이스들을 관리하는 리소스 관리자로부터 디바이스의 유형을 요청함으로써 디바이스가 다수의 스트림들을 가지는 GPU인지 여부에 엑세스할 수 있다. 각 스트림은 동작들이 순서대로 프로세싱되는 독립적인 하드웨어 큐(queue)이다.

[0046] 시스템은 디바이스에 의해 실행될 때 디바이스로 하여금 특정한 동작들을 수행하게 하는 명령어들을 생성한다(단계(208)). 특히, 명령어들은 디바이스로 하여금 서브그래프에서 각 노드에 의혀 표현되는 동작을 디바이스의 각각의 스트림에 할당하게 한다.

[0047] 예시적 시스템은 일부 하드웨어 가속기들의 계산들을 특정한 방식으로 스트림들에 할당할 수 있다(예를 들어, 하나의 동작이 스트림 A에서 실행되면, 그 후에, 관련된 동작 또한 스트림 A에서 실행되어야 함). 예를 들면, 제1 동작은 상태를 추적할 수 있고(stateful), 스트림 A에서 실행된다. 실행함으로써, 제1 동작은 제2 동작이 실행되기 전에 반드시 행해져야 하는 방식으로 하드웨어의 내부적 상태를 변경할 수 있다. 그 후, 제1 동작이 완료된 후에 제2 동작은 스트림 A에서 실행될 수 있다.

[0048] 일부 구현예에서, 2개의 내부적 하드웨어 리소스들은 동시적으로 사용될 수 없고 따라서 직렬화되어야 한다.

[0049] 일반적으로, 디바이스는 서로 종속되지 않는 동작들을 상이한 스트림들에 할당한다. 서로 종속되지 않는 동작들을 상이한 스트림들에 할당함으로써, 하드웨어는 동작이 얼마나 오래 걸릴 것인지 알 필요가 없고, 사용가능한 다수의 동작들로부터 선택하여 고비용 호스트 간섭없이 실행하기 위해 준비된 제1 동작을 실행할 수 있다.

[0050] 또한 명령어들은 디바이스로 하여금 상기 할당에 따라 서브그래프에서 노드들에 의해 표현된 동작들을 수행하게 한다. 동작들이 특정한 스트림에 할당되면, 동작들은 대기행렬에 넣어진다(queued). 디바이스는 동작들을 FIFO(first-in-firtst-out) 방식으로 수행할 수 있다. 따라서, 만약 디바이스가 단 1개의 스트림을 갖는다면,

디바이스에 할당된 동작들은 직렬적으로 수행된다. 만약 디바이스가 다수의 스트림들을 가진다면, 상이한 스트림들에서 동작들은 병렬로 수행될 수 있고, 서로에 대해 재순서화될 수 있지만, 해당 스트림 내의 동작들은 직렬적으로 수행된다. 다수의 스트림들을 사용하여 동작들을 수행하는 것은 서브그래프의 동작들을 수행하기 위한 총 시간을 감소시킨다. 이는 아래에서 도 3 및 4를 참조하여 추가로 기술된다.

[0051] 시스템은 명령어들 및 데이터를 디바이스에 제공한다(단계(210)). 일부 구현예들에서, 시스템은 디바이스에 요청을 송신하여 동작들을 시작한다. 디바이스는 요청을 수신하고, 응답으로, 시스템으로부터 수신된 명령어들을 실행한다. 예를 들면, 디바이스는 모델 입력을 수신하고, 서브그래프에서 노드들에 의해 표현된 동작들에 따라 모델 입력을 프로세싱한다.

[0052] 도 3은 가속기(302)에 의해 프로세싱되는 계산 그래프의 예시적 서브그래프(316)를 도시한다. 서브그래프(316)는 노드들(308-314)를 가지며, 각 노드는 가속기(302)에 의해 수행될 동작을 표현한다. 계산 그래프 시스템 예를 들어, 도 1의 시스템(100)은 서브그래프(316)를 가속기(302)에 할당했다.

[0053] 가속기(302)는 스트림들(304 및 305)을 가진다. 스트림들은 가속기(302)의 활용을 공유한다. GPU에서, 스트림들은 대칭적일 수 있고, 이는 모든 동작들이 임의의 스트림에서 수행될 수 있음을 의미한다. 이 대칭은 모든 가속기 디바이스에서 사용가능하지 않을 수 있다. 예를 들면, 특정 가속기 디바이스들에서 특정 스트림들은 호스트와 디바이스 메모리 사이에 데이터를 복사하는 동작들을 수행하는데 사용되어야만 할 수 있다.

[0054] 계산 그래프 시스템은 서브그래프(316)를 분석하여 어떻게 서브그래프(316)가 다수의 스트림들(304 및 306)에 할당되었는지 결정할 수 있다. 일부 구현예들에서, 시스템은 명령어들을 생성하며, 상기 명령어들은 가속기(302)로 하여금 서브그래프(316)의 노드들을 방향 에지가 연결되는 횟수를 최소화하는 방식으로 상이한 스트림들에 할당하게 한다. 스트림들 간 종속성들을 강제하는 성능 비용이 있을 수 있다. 순서화 명령어들은 일부 오버헤드 비용을 가진다. 모든 순서화 종속성은 디바이스에서 사용가능한 가능한 순서화 실행의 수를 감소시키며, 스케쥴링 유연성을 감소시킨다. 제1 스트림으로부터 방향 에지가 제2 스트림으로 연결할 때마다, 제2 스트림은 프로세싱을 완료하기 위해 제1 스트림으로부터 제2 스트림으로의 방향 에지를 가지는 동작을 위해 대기한다. 대기하는 것은 제2 스트림이 아이들링 상태에 있게 하며, 이는 GPU가 비효율적으로 활용되게 한다.

[0055] 일부 구현예들에서, 시스템은 명령어들을 생성하며, 상기 명령어들은 가속기(302)로 하여금 서브그래프(316)의 노드들을 가속기(302)의 특징들에 기초하여 할당하게 한다. 예를 들면, 가속기(302)는 고정된 수의 스트림들 즉, 스트림들(304 및 306)을 가진다. 시스템은 노드들을 할당할 수 있고, 따라서 각 스트림이 가속기(302)에 의해 유사하게 활용될 것이다. GPU들인 가속기들에 대해, 모든 스트림들은 스레드들의 단일의 커다란 풀을 공유한다.

[0056] 또한 일부 스트림들은 다른 스트림들은 수행하지 않는 특정한 동작들을 수행한다. 예를 들면, 스트림(306)은 DMA(Direct Memory Access) 동작들을 수행할 수 있지만, 스트림(304)는 수행하지 않는다. 따라서, 시스템은 각 노드를 분석하여 상기 노드에 의해 표현된 동작의 유형을 결정할 수 있고, 시스템은 노드를 동작의 상기 유형을 수행할 수 있는 스트림에 할당할 수 있다. GPU들에서, 주요 정체되는 리소스들은 데이터를 호스트들과 디바이스 메모리 간에 복사하는 DMA 엔진들이다. DMA 엔진들은 임의의 스트림에 의해 사용될 수 있다. 만약 하나의 스트림이 DMA 동작을 실행하면, 시스템은 동시적으로 계산을 실행할 수 없다. 그러므로, 예시적 시스템은 적어도 하나의 다른 스트림이 동시에 실행하기 위한 일부 계산 작업을 가지는 것을 보장한다. 시스템은 서브그래프를 식별하기 위해 분석할 수 있고, 따라서, 동작들의 할당을 관리하는 소프트웨어 모듈 또는 드라이버로 하여금 다음의 두가지 일반규칙들에 의해 노드들을 할당하게 하는 명령어들을 생성한다. 첫째, 시스템은 체인 구조에서 배열된 노드들을 동일한 스트림에 할당하려고 시도한다. 체인 구조의 노드들은 노드로부터 노드까지의 하나의 방향 에지에 따라 서로 연결된 노드들이다. 따라서, 체인에서 노드는 그것의 고유 동작을 계산하기 전에 계산을 끝내기 위한 체인에서의 이전 노드에서의 동작들을 대기해야만 한다. 노드들의 체인을 할당하는 것은 그래프에서 가지치기(branching)와 합치기(merging)이 발생하기 때문에(예를 들어, 공유된 입력 변수들 또는 공통 서브표현들로부터) 항상 가능하지는 않다.

[0057] 둘째, 시스템은 가속기(302)로 하여금 하나의 노드로부터 입력을 각각 수신하는 다수의 노드들을 고유 스트림들에 할당하게 하는 명령어들을 생성하는 것을 선택할 수 있다. 즉, 제1 노드가 다수의 상이한 노드들에 대하 다수의 출력들을 가지면, 시스템은 상기 상이한 노드들 각각을 고유 스트림에 할당한다. 상기 상이한 노드들 각각은 다른 상이한 노드들 중 임의의 노드에 대해 데이터 종속성을 가지지 않으며, 따라서, 디스조인트(disjoint) 스트림들에서 동작할 때 효율성을 개선한다.

[0058] 예시로서, 가속기(302)는 서브그래프(316)를 수신한다. 시스템에 의해 수신된 명령어들은 가속기(302)로 하여금 개시 노드(308)를 제1 스트림(306)에 할당하게 한다. 개시 노드(308)는 2개의 출력들 노드(310)에 대한 하나의 방향 에지 및 노드(314)에 대한 하나의 방향 에지를 가진다. 그러므로, 제2 규칙을 사용하여, 명령어들은 가속기(302)로 하여금 노드들(310 및 314)을 상이한 스트림들에 할당하게 한다. 또한, 노드(312)는 노드(310)의 출력만을 입력으로서 수신한다. 그러므로, 제1 규칙을 사용하여, 시스템은 노드(312)를 동일한 스트림 즉, 노드(310)과 같은 스트림(304)에 할당한다.

[0059] 상기 기술된 바와 같이, 스트림들은 동작들이 순서대로 수행되는 하드웨어 큐들이다. 따라서, 가속기(302)가 노드들을 스트림들에 할당하는 순서가 문제이다. 가속기(302)는 노드들을 스트림들에 서브그래프에서 데이터 흐름의 방향의 순서로 할당한다. 즉, 가속기(302)는 서브그래프의 하나 이상의 개시 노드들을 식별하고, 하나 이상의 개시 노드들을 할당한다. 그 후, 가속기(302)는 하나 이상의 개시 노드들의 출력들인 방향 에지들을 따라 후속 노드들을 식별하고, 가속기(302)는 상기 후속 노드들을 각각의 스트림들에 할당한다. 가속기(302)는 노드들의 할당을 서브그래프의 각 노드가 할당될 때까지 계속한다. 이 순서로 노드들을 할당하는 결과로서, 주어진 스트림 내의 동작들 또한 상기 기술된 바와 같이, 동작들이 할당된 순서로 수행될 것이다. 동작 A의 입력들이 상이한 스트림들에서 생산된 경우, 동작 A가 실행되기 전에 그들이 모두 계산되었다는 것을 보장할 필요가 있다. 동작 A 가 할당되는 스트림에서의 실행은 동작 A에 대한 모든 입력이 계산될 때까지 멈춰져야만 한다. 정확한 멈춤 메커니즘은 디바이스에 특정적이다. GPU 디바이스들에 대해, 이벤트는 입력 스트림들 각각에 대해 생성될 수 있고, 명령어들은 상기 이벤트를 신호하기 위해 각 스트림에 추가될 수 있다. 각 입력에 대해, 명령어는 또한, 동작이 실행하기 위한 관련된 이벤트에 대해 대기하게 하기 위해 A가 할당되는 스트림에 추가될 수 있다. 동작 A에 대한 입력들 중 하나 이상이 동일한 스트림에서 동작 A로서 계산되는 경우, 데이터 흐름 종속성 명령어들은 안전하게 삭제될 수 있고, 더 나은 성능으로 이끈다. 주어진 스트림 내에서, 상기 주어진 스트림에 할당된 하나 이상의 다른 노드들에 의해 표현된 동작들에 의한 입력으로서 사용되는 출력을 생성하는 주어진 스트림에 할당된 노드들에 의해 표현되는 동작들은 이미 계산되었을 것이거나 가속기(302)가 상기 하나 이상의 다른 노드들에 의해 표현된 동작들을 수행할 때 계산되도록 스케쥴링되었을 것이다.

[0060] 상기 예시를 계속하면, 스트림(304)는 데이터가 노드(310)으로부터 노드(312)로 흐르기 때문에 노드(310)에 할당되고, 그 후 노드(312)에 할당된다. 스트림에서 동작들을 실행할 때, 가속기(302)는 노드(310)에 의해 표현된 동작들을 먼저 실행하고, 그 후 노드(312)에 의해 표현된 동작들을 실행한다.

[0061] 최종 노드들 즉, 노드들(312 및 314)이 동작들을 수행한 이후에, 가속기(302)는 노드들의 출력들 또는 동작들이 완료되었다는 표시를 시스템에 리턴한다. 예시적 시스템에서, 계산 결과를 가속기(302)의 메모리로부터 호스트 메모리로 다시 복사하는 특수한 '보내기' 노드가 있고, 수신 노드에 의해 상이한 디바이스에 넘겨질 수 있거나 또는 원격 프로시저 콜(RPC, remote procedure call) 응답에서 클라이언트에 반환될 수 있다. 그 후, 시스템은, 필요시, 출력 또는 표시를 클라이언트에 리턴한다.

[0062] 노드들을 스트림들에 할당하는 다른 구현예가 도 4를 참조하여 아래에 기술될 것이다.

[0063] 도 4는 서브그래프들을 디바이스들에 할당하기 위한 예시적 프로세스(400)의 흐름도이다. 편의상, 프로세스(400)는 시스템 예를 들어 GPU에 의해 수행되는 것으로서 기술될 것이다. 예를 들면, GPU는 계산 그래프 시스템 예를 들어, 도 1의 계산 그래프 시스템(100)에 의해 생성된 명령어들을 수신할 수 있고, 상기 명령어들은 실행될 때, GPU로 하여금 프로세스(400)를 수행하게 한다.

[0064] 시스템은 노드에 의해 소비된 메모리 리소스의 양에 기초하여 또는 이전에 할당된 노드들에 의해 특정한 노드를 스트림에 할당할 수 있다. 예를 들면, 시스템은 서브그래프의 각 노드를 향하는 그리고 그로부터의 각 방향 에지에서 텐서(tensor)의 규모(dimension)을 계산할 수 있다. 상기 텐서의 규모는 동작을 수행하기 위해 디바이스에 의해 소비되는 메모리의 크기를 표시한다. 시스템은 상기 크기를 결정하기 위해 모든 텐서의 규묘를 계산할 필요가 있을 수 있다. 그 후, 시스템은 특정한 크기의 메모리를 소비하는 텐서들을 가지는 특정한 노드들을 특정한 크기의 메모리를 가지는 디바이스들에 할당할 수 있다.

[0065] 특히, 디바이스가 동작을 수행할 때, 소프트웨어 드라이버 또는 실행자는 임의의 입력들 뿐만 아니라 동작의 결과로서 계산된 임의의 출력들을 저장하기 위해 메모리를 할당한다. 디바이스 상의 메모리의 양이 제한되기 때문에, 디바이스는 메모리가 더 이상 사용되지 않는 경우 메모리를 자유화한다.

[0066] 예시로서, 시스템은 노드에 의해 표현된 동작이 특정한 스트림에서 종료되었는지 여부를 결정한다(단계(402)). 예를 들면, 시스템은 특정한 스트림에서 동작이 종료되었는지 여부를 결정하기 위해 주기적으로 스트림들을 폴

링(polling)할 수 있다. 시스템은 호스트로 하여금 스트림의 동작들의 리스트를 통해 실행이 얼마나 진행되었는지 결정하게 하는 액션을 지원할 수 있다. 일부 구현예들에서, 이벤트들 또는 마커들(markers)은 실행이 얼마나 진행되었는지 신호할 수 있다. 이벤트가 발생한 경우, 이벤트는 스트림에서 특수한 하드웨어 동작 큐에 추가될 수 있다. 호스트는 어떤 동작들이 발생했는지 결정하기 위해 이 큐를 폴링할 수 있다. 다른 스트림 구현예들은 호스트로 하여금 모든 대기행렬이 넣어진 동작들이 언제 완료되는지만 결정하게 할 수 있다. 대안적으로 또는 추가적으로, 하드웨어는 스트림이 특정 포인트에 다다르면 인터럽트(interrupt) 또는 콜백(callback)을 제공할 수 있다.

[0067] 동작이 종료된 경우, 시스템은 동작에 대한 입력들에 대해 사용된 메모리를 결정할 수 있고 다른 동작들에서의 사용을 위해 자유화될 수 있다. 시스템은 동작의 출력들을 위해 사용된 메모리를 자유화하지 않는데, 상기 출력들은 후속 노드에서 사용될 수 있기 때문이다.

[0068] 따라서, 시스템은 자유화될 소비된 메모리의 양을 결정한다(단계(404)). 시스템은 자유화될 메모리의 크기를 식별하기 위해 요청을 소프트웨어 드라이버 또는 실행자에 보낸다.

[0069] 일부 구현예들에서, 예시적 시스템은 원격 머신이 임의의 포인트의 시간에 데이터를 하드웨어 가속기의 메모리에 직접적으로 전송하는데 사용할 수 있는 RDMA(remote direct memory access) 네트워크 인터페이스들을 사용하게 한다. 이 메모리는 임의의 스트림 상에서 실행되는 임의의 다른 동작에 의해 사용되지 않아야만 한다. 예시적 시스템은 각 스트림에서의 동작들이 얼마나 진행되었는지 정확하게 알 필요가 없을 수 있다. 그러나, 시스템은 임의의 스트림에 의해 사용되지 않을 알려진 메모리를 계속 추적해야 한다. 그 후, 이 자유 메모리는 RDMA을 위해 사용될 수 있다.

[0070] 시스템은 비할당 노드들의 그룹의 각각에 대해, 상기 비할당 노드에 의해 소비된 메모리의 각각의 예측된 양을 결정한다(단계(406)). 비할당 노드들은 동작이 완료된 노드로부터의 입력들을 수신하는 노드들을 포함할 수 있다. 또한, 비할당 노드들은 동작이 완료되었지만 여전히 가속기에 의해 프로세싱되어야 하는 노드로부터 독립적인 노드들을 포함할 수 있다. 상기 기술된 바와 같이, 메모리의 예측된 양은 비할당 노드들에 대한 각각의 텐서들의 규모들을 평가함으로써 결정될 수 있다.

[0071] 시스템은 비할당 노드들의 그룹으로부터, 동작을 표현하는, 가속기에 의해 스트림 상에서 실행될 때, 자유화될 메모리의 양의 사용을 최대화하는, 제1 비할당 노드를 결정한다(단계(408)). 비할당 노드에 의해 표현된 동작을 실행하기 위해 자유화될 메모리의 양보다 많은 메모리를 요구하면, 비할당 노드는 스트림에 할당되지 않을 것이다. 제1 및 제2 동작이 자유화될 메모리의 양보다 작거나 동등한 메모리의 각각의 예측된 양을 요구하면, 시스템은 자유화될 메모리의 양의 사용을 최대화하는 동작을 선택한다. 다시 말해서, 이 경우에, 시스템은 선택된 동작을 표현하는 노드를 제1 비할당 노드로서 결정한다. 예시적 시스템은 가속기 메모리의 어떤 지역들이 일시적인 작업 공간과 동작의 출력들을 홀딩하는데 사용될 것인지 결정할 수 있을 때까지 동작을 스트림에서 대기행렬에 넣지 않는다. 메모리가 부족한 이벤트에서, 예시적 시스템은 실행하기 위해 메모리의 보다 작은 양을 요구하는 동작들을 대기행렬에 넣거나 또는 많은 입력 텐서들을 소비하여 그들이 비할당되게 할 동작들을 우선적으로 대기행렬에 넣기 위해 선택할 수 있다.

[0072] 시스템은 제1 비할당 노드에 의해 표현된 동작을 특정한 스트림에 할당한다(단계(410)). 그 후, 시스템은 특정한 스트림으로 하여금 동작을 수행하게 할 수 있고, 시스템은 도 2-3을 참조하여 상기 기술된 바와 같이 동작하는 것을 계속할 수 있다.

[0073] 본 발명의 실시예들과 본 명세서에 기술된 기능적 동작들은 본 발명에 개시된 구조들 및 그들의 구조적 균등물들 또는 그들 중 하나 이상의 조합들을 포함하는, 디지털 전자회로에서, 유형적으로 수록된 컴퓨터 소프트웨어 또는 펌웨어에서, 컴퓨터 하드웨어에서 구현될 수 있다. 본 명세서에 기술된 본 발명의 실시예들은 하나 이상의 컴퓨터 프로그램들로서 구현될 수 있다. 즉, 데이터 프로세싱 장치에 의해 실행 또는 데이터 프로세싱 장치의 동작을 제어하기 위한 유형적 비일시적인 프로그램 캐리어에 인코딩된 컴퓨터 프로그램 명령어들의 하나 이상의 모듈들. 대안적으로 또는 추가로, 프로그램 명령어들은 데이터 프로세싱 장치에 의해 실행하기 위한 적절한 수신기 장치에 전송하기 위한 정보를 인코딩하기 위해 생성된 인공적으로 생성된 전파된 신호 즉, 기계-생성 전기, 광학 또느전자기적 신호에 인코딩될 수 있다. 컴퓨터 저장 매체는 기계 판독가능 저장 디바이스, 기계 판독가능 저장 기판, 랜덤 또는 직렬 엑세스 메모리 디바이스 또는 그들 중 하나 이상의 조합일 수 있다. 그러나, 컴퓨터 저장 매체는 전파된 신호는 아니다.

[0074] 용어 “데이터 프로세싱 장치”는 예시로서 프로그래머블 프로세서, 컴퓨터, 또는 다수의 프로세서들 또는 컴퓨

터들을 포함하는 데이터를 프로세싱하기 위한 모든 종류의 장치, 디바이스들 및 기계들을 포함한다. 상기 장치는 특수 목적 논리 회로, 예를 들어 FPGA(field programmable gate array) 또는 ASIC (application specific integrated circuit)을 포함할 수 있다. 또한, 장치는 하드웨어 이외에 문제의 컴퓨터 프로그램에 대한 실행 환경을 생성하는 코드, 예를 들어 프로세서 펌웨어, 프로토콜 스택, 데이터베이스 관리 시스템, 운영 체제 또는 그들 중 하나 이상의 조합을 구성하는 코드를 포함할 수 있다.

[0075] 컴퓨터 프로그램(프로그램, 소프트웨어, 소프트웨어 어플리케이션, 모듈, 소프트웨어 모듈, 스크립트 또는 코드로도 참조되거나 기술될 수 있음)은 컴파일된 또는 인터프리트된 언어들, 또는 선언적 또는 절차적 언어들을 포함하는 임의의 형태의 프로그래밍 언어로 작성될 수 있으며, 독립 실행형 프로그램으로서 또는 모듈, 컴포넌트, 서브루틴으로서 또는 컴퓨팅 환경에서 사용하기에 적합한 기타 단위를 포함하는 임의의 형태로 배포될 수 있다. 컴퓨터 프로그램은 파일 시스템의 파일에 대응할 수 있지만, 반드시 그런 것은 아니다. 프로그램은 다른 프로그램들이나 데이터, 예를 들어, 마크업 언어 문서에 저장된 하나 이상의 스크립트들을 가지는 파일의 부분에, 문제되는 프로그램 전용 단일의 파일에 또는 다수의 조정된 파일들, 예를 들어, 하나 이상의 모듈들, 서브프로그램 또는 코드의 일부를 저장하는 파일들에 저장될 수 있다. 컴퓨터 프로그램은 하나의 컴퓨터 또는 하나의 사이트에 위치되어 있거나 다수의 사이트들에 걸쳐서 분산되어 있고 통신 네트워크에 의해 상호연결된 다수의 컴퓨터들에서 실행되도록 배포될 수 있다.

[0076] 본 명세서에서 사용된 바와 같이, “엔진” 또는 “소프트웨어 엔진”은 입력과 상이한 출력을 제공하는 소프트웨어로 구현되는 입력/출력 시스템을 지칭한다. 엔진은 라이브러리, 플랫폼, SDK(software development kit) 또는 오브젝트와 같은 기능의 인코딩된 블록일 수 있다. 각 엔진은 임의의 적절한 유형의 컴퓨팅 디바이스 예를 들어, 서버들, 모바일폰들, 태블릿 컴퓨터들, 노트북 컴퓨터들, 음악 플레이어들, 전자책 리더, 랩톱 또는 데스크톱 컴퓨터들, PDA들, 스마트폰들 또는 하나 이상의 프로세서들 및 컴퓨터 판독가능 매체를 포함하는 기타 고정식 또는 이동식 디바이스들에서 구현될 수 있다. 추가적으로, 엔진들 중 2개 이상은 동일한 컴퓨팅 디바이스 또는 상이한 컴퓨팅 디바이스들에서 구현될 수 있다.

[0077] 본 명세서에 기술된 프로세스들 및 논리 흐름들은 입력 데이터를 동작하고 출력을 생성함으로써 기능들을 수행하기 위해 하나 이상의 컴퓨터 프로그램들을 실행하는 하나 이상의 프로그래머블 컴퓨터들에 의해 수행될 수 있다. 프로세스들 및 논리 흐름들은 또한 FPGA 또는 ASIC와 같은 특수 목적 논리 회로에 의해 수행될 수 있고, 장치는 또한 특수 목적 논리 회로로서 구현될 수 있다.

[0078] 컴퓨터 프로그램의 실행에 적절한 컴퓨터들은 예시로서, 일반적 또는 특수 목적 마이크로프로세서들 또는 둘 모두, 또는 임의의 기타 종류의 중앙 프로세싱 유닛을 포함하거나 이에 기초할 수 있다. 일반적으로, 중앙 프로세싱 유닛은 읽기-전용 메모리 또는 랜덤 액세스 메모리 또는 둘 모두로부터 명령어들 및 데이터를 수신할 것이다. 컴퓨터의 필수 엘리먼트들은 명령어들을 수행하거나 실행하기 위한 중앙 프로세싱 유닛 및 명령어들 및 데이터를 저장하기 위한 하나 이상의 메모리 디바이스들이다. 일반적으로, 컴퓨터는 데이터를 저장하기 위한 하나 이상의 대형 저장 디바이스들 예를 들면, 자기적, 자기-광학 디스크들 또는 광학적 디스크들 또한 포함하거나 또는 그로부터 데이터를 수신하거나 그에 데이터를 전송하기 위해 동작적으로 결합될 수 있다. 그러나, 컴퓨터는 상기 디바이스들을 반드시 가져야하는 것은 아니다. 추가로, 컴퓨터는 다른 디바이스, 예를 들어, 몇 가지만 나열하면, 모바일 전화, 개인 휴대 정보 단말기(PDA), 모바일 오디오 또는 비디오 플레이어, 게임 콘솔, GPS 수신기 또는 휴대용 저장 디바이스 예를 들어, 범용 직렬 버스 (USB) 플래시 드라이브에 내장될 수 있다.

[0079] 컴퓨터 프로그램 명령어들 및 데이터를 저장하기에 적합한 컴퓨터 판독가능 매체는 예를 들어, EPROM, EEPROM 및 플래시 메모리 디바이스들과 같은 반도체 메모리 디바이스들; 예를 들어, 내부 하드 디스크들 또는 이동식 디스크들과 같은 자기 디스크들; 및 CD-ROM 및 DVD-ROM 디스크들을 포함하는 모든 형태의 비휘발성 메모리, 매체 및 메모리 디바이스들을 포함한다. 프로세서 및 메모리는 특수 목적 논리 회로에 의해 보충되거나 그 안에 통합될 수 있다.

[0080] 사용자와의 인터렉션을 제공하기 위해, 본 명세서에서 기술된 본 발명의 실시예들은 사용자에게 정보를 디스플레이하기 위한 디스플레이 디바이스 예를 들어, CRT(cathode ray tube) 또는 LCD(liquid crystal display) 모니터 또는 OLED 디스플레이, 및 컴퓨터에 입력을 제공하기 위한 입력 디바이스들 예를 들어, 키보드, 마우스 또는 존제 감응형 디스플레이 또는 기타 표면을 갖는 컴퓨터에서 구현될 수 있다. 다른 종류의 디바이스들도 사용자와의 인터렉션을 제공하는데 사용될 수 있다. 예를 들어, 사용자에게 제공되는 피드백은 시각 피드백, 청각 피드백 또는 촉각 피드백과 같은 임의의 형태의 감각적 피드백일 수 있고, 사용자로부터의 입력은 음향, 음성 또는 촉각 입력을 포함하는 임의의 형태로 수신될 수 있다. 추가로, 컴퓨터는 사용자에 의해 사용되는 디바이스

에 리소스들을 송수신함으로써 예를 들어, 웹 브라우저로부터 수신된 요청에 응답하여, 사용자의 사용자 디바이스상의 웹 브라우저에 웹 페이지를 전송함으로써 사용자와 인터렉션할 수 있다.

[0081] 본 명세서에서 기술된 발명의 실시예는 예를 들어 데이터 서버와 같은 백엔드 컴포넌트, 애플리케이션 서버와 같은 미들웨어 컴포넌트 또는 그래픽 사용자 인터페이스를 가지는 사용자 컴퓨터 또는 사용자가 본 명세서에 기술된 본 발명의 구현예와 인터렉션할 수 있는 웹 브라우저와 같은 프론트엔드 컴포넌트 또는 하나 이상의 상기 백엔드, 미들웨어 또는 프론트엔드 컴포넌트들의 임의의 조합을 포함하는 컴퓨팅 시스템에서 구현될 수 있다. 시스템의 컴포넌트들은 디지털 데이터 통신의 임의의 형태 또는 매체, 예를 들어 통신 네트워크에 의해 상호연결될 수 있다. 통신 네트워크들의 예시들은 LAN(local area network) 및 WAN(wide area network) 예를 들어, 인터넷을 포함한다.

[0082] 컴퓨팅 시스템은 사용자들 및 서버들을 포함할 수 있다. 사용자와 서버는 일반적으로 서로 멀리 떨어져 있으며, 일반적으로 통신 네트워크를 통해 인터렉션한다. 사용자와 서버의 관계는 각각의 컴퓨터에서 실행되고 서로 사용자-서버 관계를 갖는 컴퓨터 프로그램에 의해 발생한다.

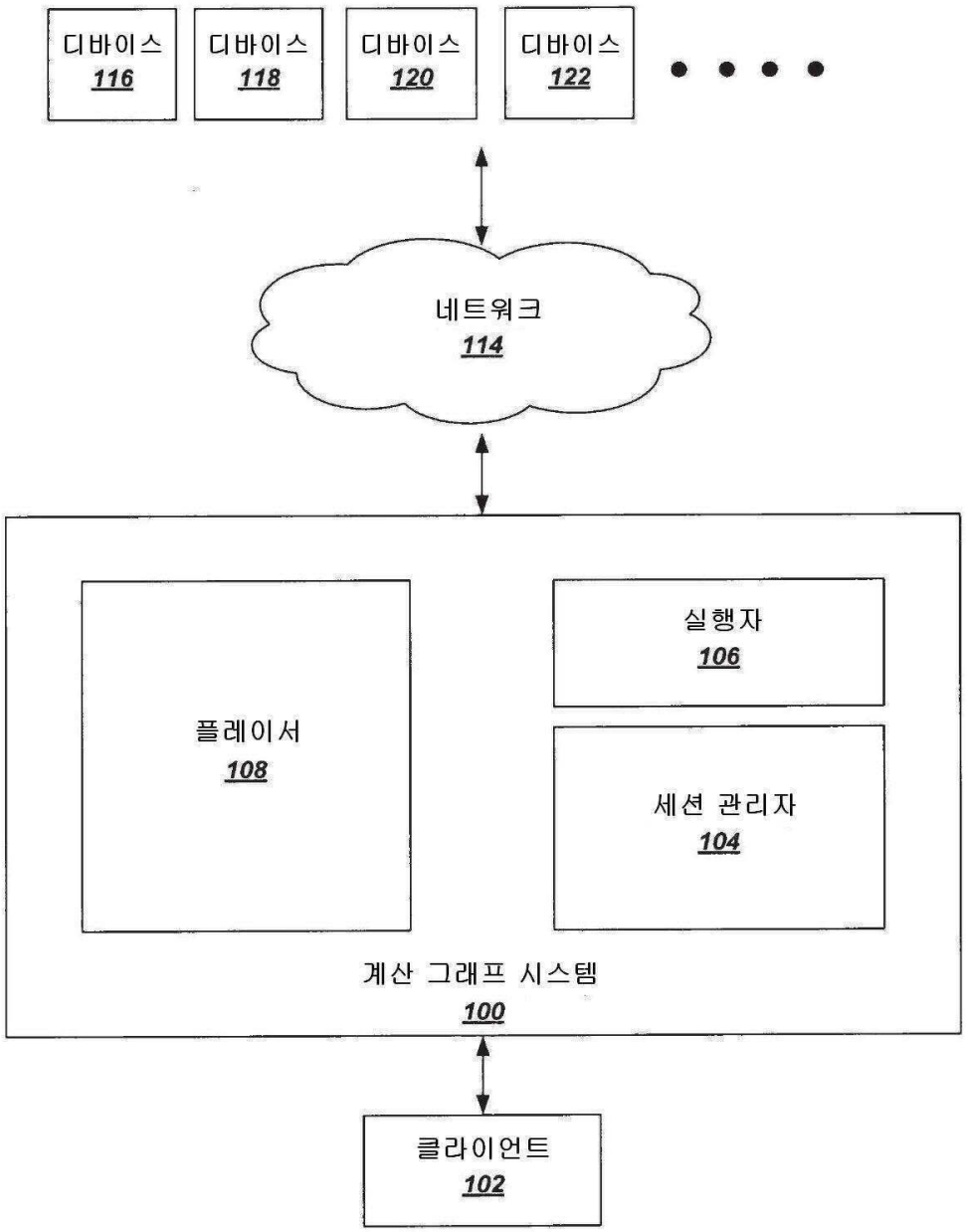
[0083] 본 명세서는 많은 특정 구현 세부내용을 포함하지만, 이들은 임의의 발명의 범위 또는 청구될 수 있는 범위에 대한 제한으로서 해석되어서는 안되며, 오히려 특정한 발명의 특정한 실시예에 특정적일 수 있는 구성들에 대한 설명으로 해석되어야 한다. 별개의 실시예의 맥락에서 본 명세서에서 기술되는 일정 구성들은 또한 단일 실시예에서 조합하여 구현될 수 있다. 반대로, 단일 실시예의 맥락에서 기술된 다양한 구성들은 또한 다수의 실시예에서 개별적으로 또는 임의의 적합한 서브 조합으로 구현될 수 있다. 게다가, 구성들은 일정 조합으로 동작하고 심지어 초기적으로 그렇게 청구되는 것으로서 상기에서 기술될 수 있지만, 청구된 조합으로부터의 하나 이상의 구성들은 일부 경우, 조합으로부터 제거될 수 있고, 청구된 조합은 서브 조합 또는 서브 조합의 변형으로 안내될 수 있다.

[0084] 유사하게, 동작들이 특정한 순서로 도면에서 도시되었지만, 이는 상기 동작들이 도시된 특정한 순서로 또는 시계열적 순서로 수행되어야 함을 요구하는 것으로서 또는 모든 도시된 동작들이 수행되어야 하는 것으로 이해되어서는 안된다. 특정 환경에서, 멀티태스킹과 병렬 프로세싱은 이점이 있다. 게다가, 상기 기술된 실시예에서 다양한 시스템 모듈들 및 컴포넌트들의 분리는 모든 실시예에서 그러한 분리가 필요한 것으로서 이해되어서는 안되며, 일반적으로 기술된 프로그램 컴포넌트들 및 시스템들은 단일의 소프트웨어 제품에 함께 통합되거나 다수의 소프트웨어 제품들에 패키징될 수 있다고 이해되어야 한다.

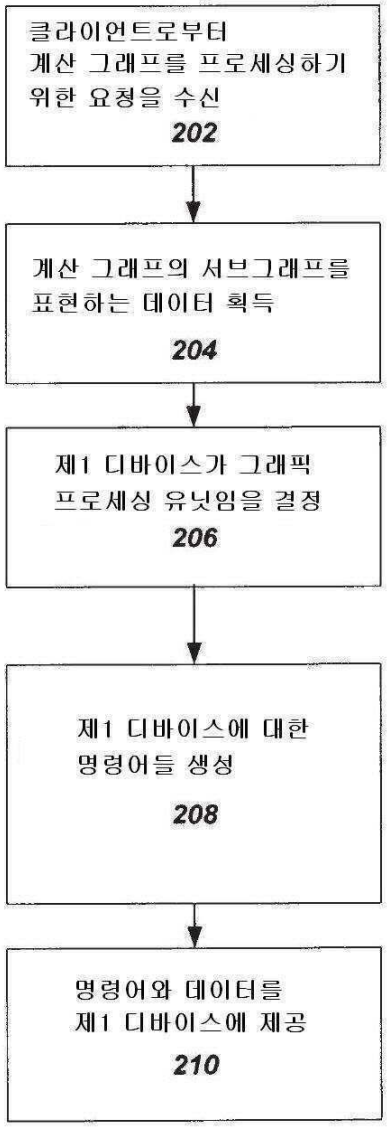
[0085] 본 발명의 특정한 실시예들이 기술되었다. 다른 실시예들도 다음의 청구항들의 범위 내에 있다. 예를 들면, 청구항들에서 기재된 액션들은 상이한 순서로 수행되고 여전히 원하는 결과들을 달성할 수 있다. 일 예시로서, 첨부 도면들에 도시된 프로세스들은 원하는 결과들을 달성하기 위해 특정한 도시된 순서, 또는 시계열적 순서를 반드시 필요로 하지 않는다. 특정 구현예에서, 멀티태스킹과 병렬 프로세싱은 이점이 있다.

도면

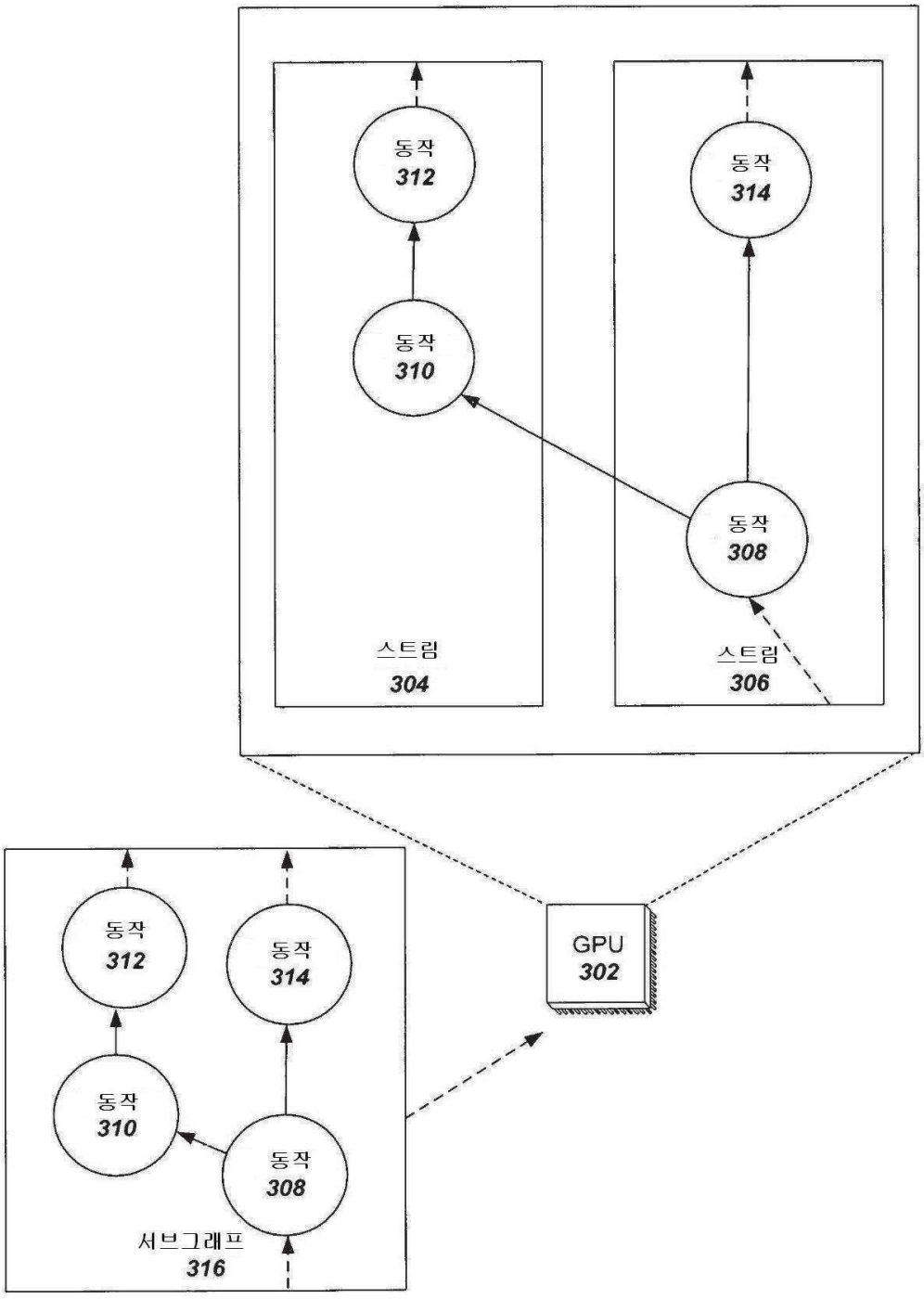
도면1


디바이스
116
디바이스
118
디바이스
120
디바이스
122
네트워크
114
플레이서
108
실행자
106
세션 관리자
104
계산 그래프 시스템
100
클라이언트
102

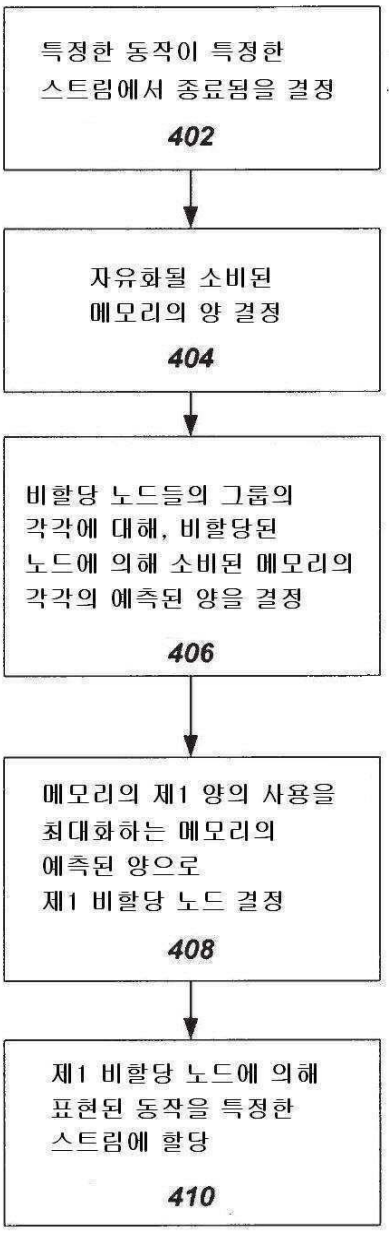
도면2


클라이언트로부터
계산 그래프를 프로세싱하기
위한 요청을 수신
202
계산 그래프의 서브그래프를
표현하는 데이터 획득
204
제1 디바이스가 그래픽
프로세싱 유닛임을 결정
206
제1 디바이스에 대한
명령어들 생성
208
명령어와 데이터를
제1 디바이스에 제공
210

도면3


동작
312
동작
310
동작
314
동작
308
스트림
304
스트림
306
GPU
302
동작
312
동작
314
동작
310
동작
308
서브그래프
316

도면4


특정한 동작이 특정한
스트림에서 종료됨을 결정
402
자유화될 소비된
메모리의 양 결정
404
비할당 노드들의 그룹의
각각에 대해, 비할당된
노드에 의해 소비된 메모리의
각각의 예측된 양을 결정
406
메모리의 제1 양의 사용을
최대화하는 메모리의
예측된 양으로
제1 비할당 노드 결정
408
제1 비할당 노드에 의해
표현된 동작을 특정한
스트림에 할당
410