(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2023-0099441
(43) 공개일자 2023년07월04일

(51) 국제특허분류(Int. Cl.)
*G06N 3/04* (2023.01) *G06F 16/182* (2019.01)
*G06N 3/08* (2023.01)
(52) CPC특허분류
*G06N 3/04* (2023.01)
*G06F 16/182* (2019.01)
(21) 출원번호 10-2021-0188781
(22) 출원일자 2021년12월27일
심사청구일자 2021년12월27일

(71) 출원인
**경희대학교 산학협력단**
경기도 용인시 기흥구 덕영대로 1732 (서천동, 경희대학교 국제캠퍼스내)
(72) 발명자
**이영구**
경기도 수원시 영통구 청명북로 33, 431동 1704호(영통동, 청명마을삼성아파트)
**칸 무하마드 누만**
경기 수원시 영통구 매영로415번길 42-23, B01호 (영통동)
**권기진**
경기도 용인시 기흥구 서그내로 23-5, 501호(서천동)
(74) 대리인
**두호특허법인**

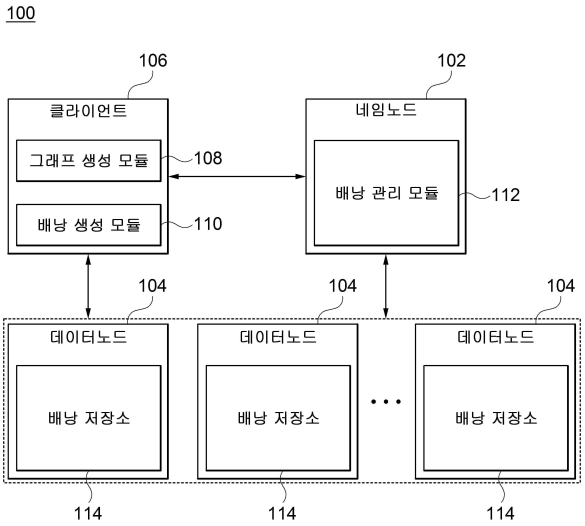
전체 청구항 수 : 총 14 항

(54) 발명의 명칭 **하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템 및 방법**

## (57) 요 약

하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템 및 방법이 개시된다. 일 실시예에 따른 그래프 신경망 처리 시스템은 학습 대상 그래프 데이터셋으로부터 복수의 서브그래프를 생성하고, 생성된 상기 복수의 서브그래프로부터 복수의 배낭(Knapsack)을 생성하는 클라이언트; 상기 복수의 배낭 각각에 포함된 서브그래프의 정보가 인덱싱된 전역 인덱스(global index)를 저장 및 관리하는 배낭 관리 모듈; 및 상기 복수의 배낭을 분산하여 저장하는 하나 이상의 배낭 저장소를 포함한다.

*대 표 도* - 도1



(52) CPC특허분류
*G06N 3/08* (2023.01)

이 발명을 지원한 국가연구개발사업

| | |
|---|---|
| 과제고유번호 | 1711134745 |
| 과제번호 | 2021-0-00859-001 |
| 부처명 | 과학기술정보통신부 |
| 과제관리(전문)기관명 | 정보통신기획평가원 |
| 연구사업명 | SW컴퓨팅산업원천기술개발(R&D,정보화) |
| 연구과제명 | 초거대 그래프의 지능적 고속 처리를 위한 그래프 DBMS 기술 개발 |
| 기 여 율 | 1/1 |
| 과제수행기관명 | 포항공과대학교 산학협력단 |
| 연구기간 | 2021.04.01 ~ 2021.12.31 |

# 명 세 서

## 청구범위

### 청구항 1

학습 대상 그래프 데이터셋으로부터 복수의 서브그래프를 생성하고, 생성된 상기 복수의 서브그래프로부터 복수의 배낭(Knapsack)을 생성하는 클라이언트;

상기 복수의 배낭 각각에 포함된 서브그래프의 정보가 인덱싱된 전역 인덱스(global index)를 저장 및 관리하는 배낭 관리 모듈; 및

상기 복수의 배낭을 분산하여 저장하는 하나 이상의 배낭 저장소를 포함하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템.

### 청구항 2

청구항 1에 있어서,

상기 클라이언트는,

상기 학습 대상 그래프 데이터셋으로부터 상기 복수의 서브그래프를 생성하는 그래프 생성 모듈; 및

생성된 상기 복수의 서브그래프로부터 상기 복수의 배낭(Knapsack)을 생성하는 배낭 생성 모듈을 포함하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템.

### 청구항 3

청구항 2에 있어서,

상기 배낭 생성 모듈은,

기 설정된 최대 허용 한계 내에서 상기 복수의 서브그래프를 그룹화하여 상기 복수의 배낭을 생성하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템.

### 청구항 4

청구항 3에 있어서,

상기 배낭 생성 모듈은,

입력된 서브그래프의 크기가 현재 생성중인 배낭의 가용 용량보다 큰 경우, 상기 현재 생성중인 배낭을 상기 배낭 저장소에 저장하고, 새로운 배낭을 생성하여 상기 입력된 서브그래프를 저장하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템.

### 청구항 5

청구항 3에 있어서,

상기 배낭 생성 모듈은,

입력된 서브그래프의 크기가 상기 배낭의 최대 허용 한계를 초과하는 경우, 상기 입력된 서브그래프를 복수의 청크로 분할하고, 분할된 상기 복수의 청크 각각을 복수의 배낭에 저장하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템.

### 청구항 6

청구항 5에 있어서,

상기 배낭 생성 모듈은,

상기 복수의 배낭 각각의 헤더 섹션에, 해당 배낭이 보유한 청크의 다음 청크를 보유한 배낭의 식별 정보를 기

록하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템.

**청구항 7**

청구항 2에 있어서,

상기 배낭 생성 모듈은,

생성된 배낭에 포함된 각 서브그래프의 식별 정보, 시작 오프셋 및 종료 오프셋을 포함하는 인덱스 테이블을 해당 배낭의 헤더 섹션에 기록하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템.

**청구항 8**

청구항 2에 있어서,

상기 배낭 저장소는,

저장 대상 배낭을 수신하고, 수신된 배낭에 대한 인덱스 레코드를 생성하여 상기 글로벌 인덱스에 인덱싱하며, 상기 수신된 배낭을 스토리지에 저장하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템.

**청구항 9**

학습 대상 그래프 데이터셋으로부터 복수의 서브그래프를 생성하는 단계;

생성된 상기 복수의 서브그래프로부터 복수의 배낭(Knapsack)을 생성하는 단계;

상기 복수의 배낭 각각에 포함된 서브그래프의 정보를 전역 인덱스(global index)에 인덱싱하는 단계; 및

상기 복수의 배낭을 분산하여 저장하는 단계를 포함하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 방법.

**청구항 10**

청구항 9에 있어서,

상기 배낭 생성 단계는,

기 설정된 최대 허용 한계 내에서 상기 복수의 서브그래프를 그룹화하여 상기 복수의 배낭을 생성하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 방법.

**청구항 11**

청구항 10에 있어서,

상기 배낭 생성 단계는,

입력된 서브그래프의 크기가 현재 생성중인 배낭의 가용 용량보다 큰 경우, 상기 현재 생성중인 배낭을 상기 배낭 저장소에 저장하고, 새로운 배낭을 생성하여 상기 입력된 서브그래프를 저장하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 방법.

**청구항 12**

청구항 10에 있어서,

상기 배낭 생성 단계는,

입력된 서브그래프의 크기가 상기 배낭의 최대 허용 한계를 초과하는 경우, 상기 입력된 서브그래프를 복수의 청크로 분할하고, 분할된 상기 복수의 청크 각각을 복수의 배낭에 저장하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 방법.

**청구항 13**

청구항 12에 있어서,

상기 배낭 생성 단계는,

상기 복수의 배낭 각각의 헤더 섹션에, 해당 배낭이 보유한 청크의 다음 청크를 보유한 배낭의 식별 정보를 기록하는 단계를 더 포함하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 방법.

**청구항 14**

청구항 9에 있어서,

상기 배낭 생성 단계는,

생성된 배낭에 포함된 각 서브그래프의 식별 정보, 시작 오프셋 및 종료 오프셋을 포함하는 인덱스 테이블을 해당 배낭의 헤더 섹션에 기록하는 단계를 더 포함하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 방법.

## *발명의 설명*

### *기 술 분 야*

[0001] 개시되는 실시예들은 그래프 신경망을 학습하고 분석하기 위한 기술과 관련된다.

### *배 경 기 술*

[0002] 그래프 신경망(Graph Neural Network, GNN)은 그래프 데이터의 모델링 및 분석을 위한 최적의 딥 러닝 프레임워크 솔루션 중 하나이다. GNN은 뛰어난 표현력과 인상적인 성능으로 인해 소셜 네트워크 분석, 추천 시스템, 이상 탐지 등과 같은 많은 산업 응용 프로그램에서 널리 사용된다.

[0003] 그래프의 용량이 증가하면서 GNN을 이용하여 이를 처리하는데 많은 어려움이 존재한다. 예를 들어, 소셜 네트워크 그래프에는 약 20억 개 이상의 노드와 약 1조 개 이상의 에지가 존재할 수 있으며, 이를 종합하면 100TB 이상의 데이터 용량을 가질 수 있다. 이 정도 규모의 그래프 데이터에 대한 학습은 단일 컴퓨터를 통해 이루어질 수 없다. 다시 말해 실제 산업에 적용될 수 있는 규모의 문제를 해결하기 위해서는 확장성, 내결함성, 완전한 기능을 갖추고 학습/추론 워크로드의 통합성을 갖춘 학습 시스템을 구축해야 한다. 특히 GNN의 학습 내지 추론을 위해 사용되는 데이터는 그 종속성으로 인해 일반적인 데이터와는 그 특성이 다르다. 예를 들어 데이터가 병렬로 처리되는 기존의 서버 프레임워크에서 각 샘플의 계산 결과는 다른 샘플과 독립적이지만, GNN의 경우 각 샘플의 계산 결과는 이웃 샘플의 결과에 종속된다. 즉 GNN 학습 및 추론을 위해 사용되는 데이터 뭉치는 서브그래프 형태이며 이 서브그래프를 어떻게 구성하는지에 GNN 학습과 추론 성능이 좌우된다.

[0004] 이에 따라 최근에는 하둡 에코 시스템(Hadoop echo system)을 활용한 확장 가능한 시스템을 이용하여 GNN 모델에 대한 학습 및 추론을 수행하기 위한 방법들이 제안되었다. 제안된 시스템에서는 하둡 분산 파일시스템(HDFS, Hadoop Distributed File System)을 이용하여 GNN 모델을 훈련하였다. 그러나 기존의 HDFS를 GNN 모델의 학습에 곧바로 적용할 경우 여러 문제가 발생한다.

[0005] 이는 HDFS가 주로 GB 또는 TB 단위의 대용량 파일을 처리하도록 설계된 반면, GNN 모델의 학습에 사용되는 서브그래프(k-hop subgraph)들은 많은 수의 작은 파일로 구성되기 때문이다. HDFS는 네임 노드(NameNode)라는 마스터 노드와 데이터 노드(DataNode)라는 슬레이브 노드로 구성되는데, GNN 모델의 경우 각 서브그래프들이 별도의 파일로 저장되므로, 네임 노드의 메모리가 감당할 수 없을 정도로 파일을 가진 경우가 많아 HDFS가 제대로 작동하기 어렵다. 또한 GNN 응용프로그램은 그 특성상 파일을 자주 읽어들여야 하는데, 작언 파일에 자주 액세스하면 데이터 노드간에 너무 많은 쿼리와 홉핑(hopping)이 발생하여 I/O 성능이 심각하게 떨어지게 된다.

## *발명의 내용*

### *해결하려는 과제*

[0006] 개시되는 실시예들은 HDFS에서 GNN 그래프의 학습 및 추론을 효과적으로 수행하기 위한 기술적인 수단을 제공하기 위한 것이다.

### *과제의 해결 수단*

[0007] 예시적인 실시예에 따르면, 학습 대상 그래프 데이터셋으로부터 복수의 서브그래프를 생성하고, 생성된 상기 복수의 서브그래프로부터 복수의 배낭(Knapsack)을 생성하는 클라이언트; 상기 복수의 배낭 각각에 포함된 서브그래프의 정보가 인덱싱된 전역 인덱스(global index)를 저장 및 관리하는 배낭 관리 모듈; 및 상기 복수의 배낭

을 분산하여 저장하는 하나 이상의 배낭 저장소를 포함하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템이 제공된다.

[0008] 상기 클라이언트는, 상기 학습 대상 그래프 데이터셋으로부터 상기 복수의 서브그래프를 생성하는 그래프 생성 모듈; 및 생성된 상기 복수의 서브그래프로부터 상기 복수의 배낭(Knapsack)을 생성하는 배낭 생성 모듈을 포함할 수 있다.

[0009] 상기 배낭 생성 모듈은, 기 설정된 최대 허용 한계 내에서 상기 복수의 서브그래프를 그룹화하여 상기 복수의 배낭을 생성할 수 있다.

[0010] 상기 배낭 생성 모듈은, 입력된 서브그래프의 크기가 현재 생성중인 배낭의 가용 용량보다 큰 경우, 상기 현재 생성중인 배낭을 상기 배낭 저장소에 저장하고, 새로운 배낭을 생성하여 상기 입력된 서브그래프를 저장할 수 있다.

[0011] 상기 배낭 생성 모듈은, 입력된 서브그래프의 크기가 상기 배낭의 최대 허용 한계를 초과하는 경우, 상기 입력된 서브그래프를 복수의 청크로 분할하고, 분할된 상기 복수의 청크 각각을 복수의 배낭에 저장할 수 있다.

[0012] 상기 배낭 생성 모듈은, 상기 복수의 배낭 각각의 헤더 섹션에, 해당 배낭이 보유한 청크의 다음 청크를 보유한 배낭의 식별 정보를 기록할 수 있다.

[0013] 상기 배낭 생성 모듈은, 생성된 배낭에 포함된 각 서브그래프의 식별 정보, 시작 오프셋 및 종료 오프셋을 포함하는 인덱스 테이블을 해당 배낭의 헤더 섹션에 기록할 수 있다.

[0014] 상기 배낭 저장소는, 저장 대상 배낭을 수신하고, 수신된 배낭에 대한 인덱스 레코드를 생성하여 상기 글로벌 인덱스에 인덱싱하며, 상기 수신된 배낭을 스토리지에 저장할 수 있다.

[0015] 다른 예시적인 실시예에 따르면, 학습 대상 그래프 데이터셋으로부터 복수의 서브그래프를 생성하는 단계; 생성된 상기 복수의 서브그래프로부터 복수의 배낭(Knapsack)을 생성하는 단계; 상기 복수의 배낭 각각에 포함된 서브그래프의 정보를 전역 인덱스(global index)에 인덱싱하는 단계; 및 상기 복수의 배낭을 분산하여 저장하는 단계를 포함하는, 하둡 분산 파일시스템 기반의 그래프 신경망 처리 방법이 제공된다.

[0016] 상기 배낭 생성 단계는, 기 설정된 최대 허용 한계 내에서 상기 복수의 서브그래프를 그룹화하여 상기 복수의 배낭을 생성할 수 있다.

[0017] 상기 배낭 생성 단계는, 입력된 서브그래프의 크기가 현재 생성중인 배낭의 가용 용량보다 큰 경우, 상기 현재 생성중인 배낭을 상기 배낭 저장소에 저장하고, 새로운 배낭을 생성하여 상기 입력된 서브그래프를 저장할 수 있다.

[0018] 상기 배낭 생성 단계는, 입력된 서브그래프의 크기가 상기 배낭의 최대 허용 한계를 초과하는 경우, 상기 입력된 서브그래프를 복수의 청크로 분할하고, 분할된 상기 복수의 청크 각각을 복수의 배낭에 저장할 수 있다.

[0019] 상기 배낭 생성 단계는, 상기 복수의 배낭 각각의 헤더 섹션에, 해당 배낭이 보유한 청크의 다음 청크를 보유한 배낭의 식별 정보를 기록하는 단계를 더 포함할 수 있다.

[0020] 상기 배낭 생성 단계는, 생성된 배낭에 포함된 각 서브그래프의 식별 정보, 시작 오프셋 및 종료 오프셋을 포함하는 인덱스 테이블을 해당 배낭의 헤더 섹션에 기록하는 단계를 더 포함할 수 있다.

## 발명의 효과

[0021] 개시되는 실시예들에 따르면 그래프로부터 생성된 다수의 서브그래프를 배낭(Knapsack) 구조로 그룹화함으로써 HDFS의 병목 지점인 I/O의 횟수를 효과적으로 절감할 수 있으며, 전역 인덱스 테이블을 이용하여 배낭에 저장된 서브그래프를 용이하게 탐색할 수 있다. 이에 따라 개시되는 실시예들에 따르면 HDFS를 이용하여 효과적으로 GNN 그래프의 학습 및 추론을 수행할 수 있다.

## 도면의 간단한 설명

[0022] 도 1은 일 실시예에 따른 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템(100)을 설명하기 위한 블록도

도 2는 일 실시예에 따른 배낭의 구조를 설명하기 위한 블록도

도 3은 도 2에 도시된 구조에 따라 복수의 서브그래프가 저장된 배낭을 예시한 예시도

도 4는 일 실시예에 따른 배낭 생성 모듈(110)에서 배낭을 생성하는 과정을 알고리즘 형태로 나타낸 프로그램 코드

도 5는 일 실시예에 따른 배낭 저장소(114)에서 배낭을 인덱싱하고 저장하는 과정을 알고리즘 형태로 나타낸 프로그램 코드

도 6은 일 실시예에 따른 그래프 신경망 처리 시스템(100)에서 원하는 배낭을 추출하는 과정을 알고리즘 형태로 나타낸 프로그램 코드

도 7은 개시되는 실시예들에 따른 효과를 설명하기 위한 그래프

도 8은 일 실시예에 따른 하둡 분산 파일시스템 기반의 그래프 신경망 처리 방법(800)을 설명하기 위한 흐름도

## *발명을 실시하기 위한 구체적인 내용*

[0023] 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.

[0024] 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, 후술되는 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.

[0025] 도 1은 일 실시예에 따른 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템(100)을 설명하기 위한 블록도이다.

[0026] 하둡 분산 파일시스템(HDFS, Hadoop Distributed File System)은 가장 널리 사용되는 오픈 소스 기반의 분산 파일 시스템 중 하나이다. 도 1에 도시된 바와 같이, HDFS는 네임 노드(NameNode, 102), 하나 이상의 데이터 노드(DataNode, 104) 및 클라이언트(Client, 106)로 구성된다.

[0027] 네임 노드(102)는 파일 시스템 네임스페이스와 메타데이터를 관리하고 클라이언트(106)의 다양한 요청에 응답하는 역할을 한다. 데이터 노드(104)는 데이터 저장에 사용되며 시스템 가용성을 보장하기 위해 데이터 복제 메커니즘을 제공한다. 데이터노드(104)에 저장될 파일은 클라이언트(106)에 의해 블록으로 나뉜다. 네임노드(102)는 각 블록을 저장할 데이터노드(104)를 할당하고, 메모리에 각 블록의 위치 정보를 포함하는 메타데이터를 생성한다. 클라이언트(106)는 네임노드(102)에 의해 할당된 데이터노드(104)에 블록을 저장한다. 데이터 읽기의 경우 클라이언트(106)는 네임노드(102)에 읽기 요청을 보내고, 네임노드(102)는 실제 데이터에 대한 메타데이터를 반환한다. 클라이언트(106)는 상기 메타데이터를 기반으로 데이터노드(104)에서 파일의 각 블록을 읽어들인다. 마지막으로 모든 파일 블록은 클라이언트(106)에서 완전한 파일로 어셈블된다.

[0028] HDFS를 이용하여 그래프 데이터셋을 처리할 경우의 문제는 HDFS의 읽기, 쓰기 프로세스가 다수의 작은 파일에 적합하지 않다는 것이다. 예를 들어 하나의 메타데이터의 크기가 약 150바이트라고 가정할 경우, 1억 개의 파일은 150기가바이트의 메타데이터를 생성한다. 그러나 150기가바이트 용량의 메타데이터를 메모리상에서 관리할 수 있는 네임노드는 현실적으로 존재하지 않는다. 이와 같은 문제를 해결하기 위해, 개시되는 실시예에 따른 그래프 신경망 처리 시스템(100)은 다음과 같은 구조를 포함하도록 구성된다.

[0029] 먼저, 클라이언트(106)는 학습 대상 그래프 데이터셋으로부터 복수의 서브그래프를 생성하고, 생성된 상기 복수의 서브그래프로부터 복수의 배낭(Knapsack)을 생성하도록 구성된다.

[0030] 구체적으로 클라이언트(106)는 그래프 생성 모듈(108) 및 배낭 생성 모듈(110)을 포함한다.

[0031] 그래프 생성 모듈(108)은 학습 대상 그래프 데이터셋으로부터 신경망 아키텍처의 요구 사항에 따라 복수의 서브

그래프(k-hop subgraph)를 생성한다. GNN 학습 및 추론을 위하여 그래프 데이터셋으로부터 복수의 서브그래프를 생성하는 방법은 본 기술분야에서 다양한 방법이 알려져 있으므로 여기서는 이에 대한 자세한 설명을 생략하기로 한다.

[0032] 배낭 생성 모듈(110)은 생성된 상기 복수의 서브그래프로부터 복수의 배낭(Knapsack)을 생성한다. 일 실시예에서, 배낭 생성 모듈(110)은 기 설정된 최대 허용 한계 내에서 복수의 서브그래프를 그룹화하여 복수의 배낭을 생성할 수 있다. 즉, 배낭 생성 모듈(110)은 허용 한계에 도달할 때까지 입력된 서브그래프를 배낭에 추가한 뒤 새로운 빈 배낭을 생성하게 된다. 이때 상기 최대 허용 한계는 HDFS의 블록 크기와 동일하게 설정될 수 있다.

[0033] 새로운 서브그래프를 입력받으면, 배낭 생성 모듈(110)은 먼저 입력된 서브그래프의 크기를 계산한다. 만약 입력된 서브그래프의 크기가 현재 생성중인 배낭의 가용 용량보다 큰 경우, 배낭 생성 모듈(1080은 현재 생성중인 배낭을 후술할 배낭 저장소(114)에 저장하고, 새로운 배낭을 생성하여 상기 입력된 서브그래프를 저장한다.

[0034] 만약 입력된 서브그래프의 크기가 상기 배낭의 최대 허용 한계를 초과하는 경우, 다시 말해 서브그래프를 하나의 배낭에 수용할 수 없는 경우, 배낭 생성 모듈(110)은 입력된 서브그래프를 복수의 청크로 분할하고, 분할된 상기 복수의 청크 각각을 복수의 배낭에 나누어 저장한다. 이 경우 상기 복수의 청크는 현재 생성중인 배낭의 남은 용량을 기반으로 생성될 수 있다. 예를 들어 sk가 현재 배낭의 남은 용량인 경우, 서브그래프는 c = {c1, c2, ..., cn} 크기의 청크를 생성하는 방식으로 분할될 수 있다. 여기서 첫 번째 청크의 크기인 c1은 현재 생성중인 배낭에 남아있는 용량의 크기인 나와 같도록 설정되고, 나머지 청크 {c2, ..., cn-1}은 HDFS 블록 크기와 동일한 크기로 설정될 수 있다. 마지막 청크 cn은 서브그래프의 전체 용량에 따라 이전 청크와 크기가 같을 수도 있고 아닐 수도 있다.  청크가 생성된 후 첫 번째 청크 c1이 현재 배낭에 추가되고 현재 배낭이 저장 및 인덱싱을 위해 푸시된다. 나머지 청크는 별도의 배낭에 저장되고 인덱싱된다.

[0035] 도 2는 일 실시예에 따른 배낭의 구조를 설명하기 위한 블록도이고, 도 3은 도 2의 구조에 따라 복수의 서브그래프가 저장된 배낭을 예시한 예시도이다. 각 배낭은 헤더 섹션과 데이터 섹션으로 나뉘어진다. 헤더 섹션에는 배낭의 식별 정보(ID), 배낭에 포함된 서브그래프의 개수, 부분 필드(partial field) 및 인덱스 테이블이 포함된다.

[0036] 만약 배낭에 특정 서브그래프의 분할된 청크가 저장되어 있는 경우 부분 필드에는 해당 서브그래프의 나머지 청크를 보유하는 다음 배낭의 식별 정보가 기록된다. 만약 배낭이 분할되지 않은 전체 서브그래프만을 포함하는 경우 부분 필드에는 아무 것도 기록되지 않는다. 즉 서브그래프의 용량이 배낭의 최대 허용 한계를 초과하여 복수의 청크로 분할된 경우, 배낭 생성 모듈(110)은 각 청크를 저장하는 복수의 배낭 각각의 헤더 섹션에, 해당 배낭이 보유한 청크의 다음 청크를 보유한 배낭의 식별 정보를 기록하게 된다. 이와 같은 과정을 통해 동일한 서브그래프의 각 청크를 보유하는 배낭들은 링크드 리스트(linked list) 방식으로 연결된다.

[0037] 인덱스 테이블은 배낭에 포함된 각 서브그래프의 식별 정보, 시작 오프셋 및 종료 오프셋을 포함한다. 이는 배낭 생성 과정에서 배낭 생성 모듈(110)에 의해 기록될 수 있다. 도 4는 일 실시예에 따른 배낭 생성 모듈(110)에서 배낭을 생성하는 과정을 알고리즘 형태로 나타낸 것이다.

[0038] 네임노드(102)에는 배낭 관리 모듈(112)을 포함한다. 배낭 관리 모듈(112)은 배낭 생성 모듈(110)에서 생성한 복수의 배낭 각각에 포함된 서브그래프의 정보가 인덱싱된 전역 인덱스(global index)를 저장 및 관리한다.

[0039] 마지막으로 각 데이터노드(104)는 상기 복수의 배낭을 분산하여 저장하기 위한 배낭 저장소(114)를 포함한다. 클라이언트(106)로부터 배낭이 저장 대상 배낭이 수신되면, 배낭 저장소(114)는 수신된 배낭에 대한 인덱스 레코드를 생성하여 글로벌 인덱스에 인덱싱하고 배낭을 저장한다. 상기 인덱스 레코드에는 배낭의 식별 정보, 포함된 서브그래프 수, 부분 필드, 포함된 각 서브그래프의 시작 오프셋 및 종료 오프셋, 그리고 해당 배낭이 저장된 HDFS 내 위치 정보 등이 포함될 수 있다. 도 5는 일 실시예에 따른 배낭 저장소(114)에서 배낭을 인덱싱하고 저장하는 과정을 알고리즘 형태로 나타낸 것이다.

[0040] 상술한 과정을 통해 저장된 서브그래프에 액세스하기 위한 과정을 설명하면 다음과 같다. 먼저 클라이언트(106)는 전역 인덱스에서 희망하는 정점(vertex)의 정보를 조회한다. 해당 정점을 포함하는 인덱스 레코드가 발견되면, 클라이언트(106)는 해당 정점이 포함된 데이터노드(104)의 HDFS 경로를 인덱스 레코드에서 가져온다. 이후 클라이언트(106)는 가져온 경로를 이용하여 데이터노드(104)에서 실제 배낭을 읽는다. 읽어드린 배낭이 큰 서브그래프의 일부를 포함하는 부분 배낭인 경우, 클라이언트(106)는 배낭의 부분 필드를 이용하여 서브그래프의 나머지 부분을 포함하는 다른 모든 배낭을 읽고 각 배낭에 저장된 청크를 결합한다. 그런 다음 클라이언트(106)는 배낭의 헤더에서 배낭 내부의 정점 정보를 읽어들이고, 해당 정점의 서브그래프를 배낭에서 검색하여

훈련을 수행한다. 이 과정은 원하는 모든 정점이 처리될 때까지 반복 수행된다. 도 6은 일 실시예에 따른 그래프 신경망 처리 시스템(100)에서 원하는 배낭을 추출하는 과정을 알고리즘 형태로 나타낸 것이다.

[0041] 도 7은 개시되는 실시예들에 따른 효과를 설명하기 위한 그래프이다. 구체적으로, 도 7의 그래프는 HDFS에서 서브그래프를 개별적으로 읽어들일 때와 배낭을 이용할 경우의 HDFS 읽기 시간을 비교한 것이다. 도시된 바와 같이, 배낭을 이용하지 않을 경우는 노드의 증가에 따라 개별 서브그래프들을 읽어들이는 시간이 선형적으로 증가하는 반면, 배낭을 이용하면 서브그래프의 로드 시간이 매우 작게 유지된다. 그 이유는 HDFS 쓰기 메카니즘 때문이다. 개별 하위 그래프를 사용하면 큰 단점이 있다. 첫째, 대규모 데이터 세트에는 수백만 ~ 수십억 개의 노드가 포함되어 있으며 모든 노드에 대해 서브그래프를 저장하면 엄청난 양의 작은 파일 또는 순서 n, h가 생성된다. 여기서 n은 데이터 세트의 노드 수이고 h는 홉 수이다. 이러한 엄청난 양의 파일은 HDFS 저장에 매우 큰 장애물이 된다. 둘째, 이렇게 많은 수의 그래프를 로드하면 읽기 시간이 늘어날 뿐만 아니라 네트워크 트래픽도 늘어난다. 대조적으로, GNN의 요구 사항에 따라 이러한 서브그래프를 배낭 형태로 그룹화하면 앞서 언급한 문제를 해결할 뿐만 아니라 서브그래프 로딩 시간을 줄여 GNN 학습 시간을 크게 감소시킨다.

[0042] 도 8은 일 실시예에 따른 하둡 분산 파일시스템 기반의 그래프 신경망 처리 방법(800)을 설명하기 위한 흐름도이다. 도시된 방법은 하나 이상의 프로세서들, 및 상기 하나 이상의 프로세서들에 의해 실행되는 하나 이상의 프로그램들을 저장하는 메모리를 구비한 컴퓨팅 장치, 예컨대 전술한 도 8은 그래프 신경망 처리 시스템(100)에 의해 수행될 수 있다. 도시된 흐름도에서는 상기 방법 또는 과정을 복수 개의 단계로 나누어 기재하였으나, 적어도 일부의 단계들은 순서를 바꾸어 수행되거나, 다른 단계와 결합되어 함께 수행되거나, 생략되거나, 세부 단계들로 나뉘어 수행되거나, 또는 도시되지 않은 하나 이상의 단계가 부가되어 수행될 수 있다.

[0043] 단계 802에서, 그래프 신경망 처리 시스템(100)의 클라이언트(106)는 학습 대상 그래프 데이터셋으로부터 복수의 서브그래프를 생성한다.

[0044] 단계 804에서, 클라이언트(106)는 생성된 상기 복수의 서브그래프로부터 복수의 배낭(Knapsack)을 생성한다.

[0045] 단계 806에서, 클라이언트(106)는 상기 복수의 배낭 각각에 포함된 서브그래프의 정보를 전역 인덱스(global index)에 인덱싱한다.

[0046] 단계 808에서, 클라이언트(106)는 상기 복수의 배낭을 데이터노드(104)에 분산하여 저장한다.

[0047] 이상에서 대표적인 실시예를 통하여 본 발명에 대하여 상세하게 설명하였으나, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 전술한 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 특허청구범위뿐만 아니라 이 특허청구범위와 균등한 것들에 의해 정해져야 한다.

## *부호의 설명*

[0048] 100: 하둡 분산 파일시스템 기반의 그래프 신경망 처리 시스템

102: 네임노드

104: 데이터노드

106: 클라이언트

108: 그래프 생성 모듈

110: 배낭 생성 모듈

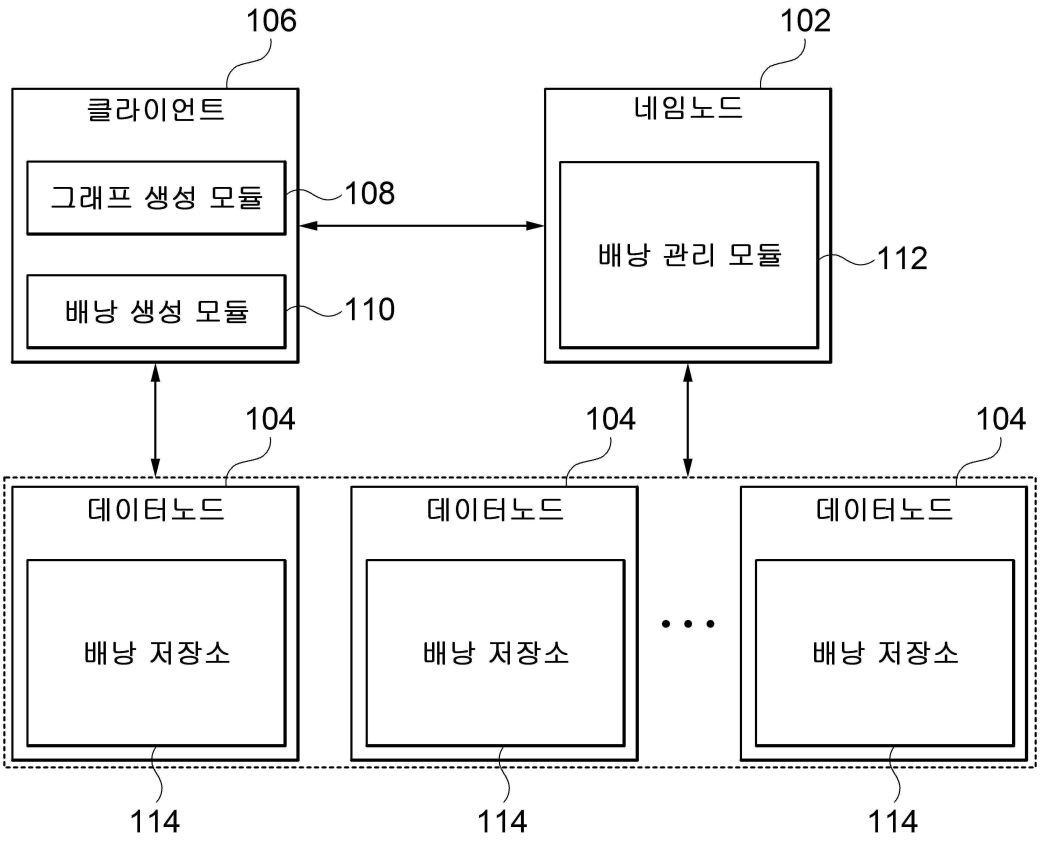
112: 배낭 관리 모듈

114: 배낭 저장소

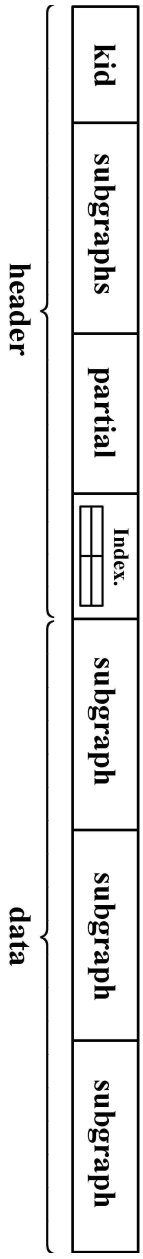
도면

도면1

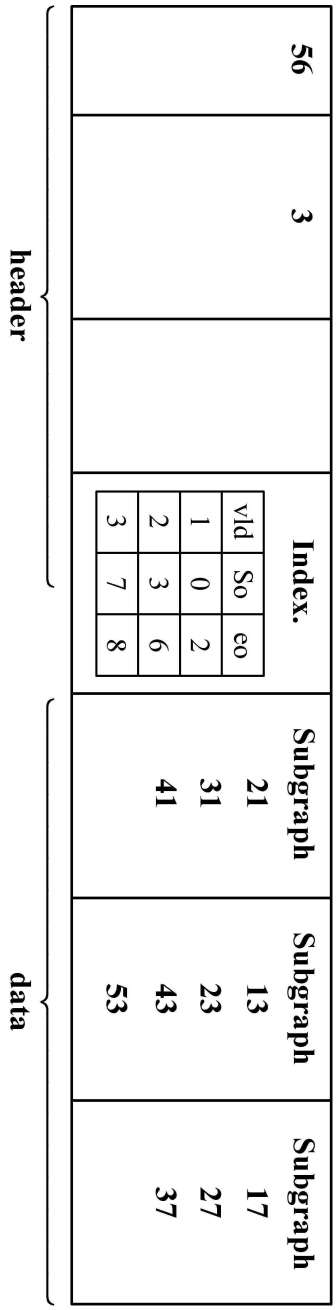
100


106
클라이언트
그래프 생성 모듈
108
배낭 생성 모듈
110
102
네임노드
배낭 관리 모듈
112
104
104
104
데이터노드
데이터노드
데이터노드
배낭 저장소
배낭 저장소
배낭 저장소
114
114
114

도면2


header
data
kid
subgraphs
partial
Index.
subgraph
subgraph
subgraph

도면3


header
data
56
3
Index.
vld
So
eo
1
0
2
2
3
6
3
7
8
Subgraph
21
31
41
Subgraph
13
23
43
53
Subgraph
17
27
37

도면4

**Algorithm 1** Knapsack Generation

**Input:** Subgraph, Path
**Output:** Knapsack

```
1: function GenerateKnapsack(subgraph, path)
2:     knapsack ← GetOrCreateKnapsack()
3:     sgSize ← EstimateSize(subgraph)
4:     kID ← GetLastKnapsackId()
5:     if (Size(sgSize) > HDFS Block Size) then
6:         emptySize ← GetEmptySpace(knapsack)
7:         chuncks ← CreateChuncks(subgraph, emptySize)
8:         for each chunk ∈ chunks do
9:             if knapsack.subgraphs = 0 then
10:                kID = kID + 1
11:                knapsack.id ← kID
12:                knapsack.data ← Append(chunk)
13:            else
14:                knapsack.data ← Add(chunk)
15:            end if
16:            iOffset ← knapsack.Index.LastOffset
17:            knapsack.subgraphs ← knapsack.subgraphs + 1
18:            knapsack.partial ← kID + 1
19:            knapsack.UpdateIndex(subgraph.nID, iOffset + 1, iOffset + Size(chunk))
20:            StoreKnapsack(knapsack, path)
21:            knapsack ← CreateKnapsack()
22:        end for
23:    else
24:        kpSize ← GetSize(knapsack)
25:        if (kpSize + sgSize > HDFS Block Size) then
26:            StoreKnapsack(knapsack, path)
27:            knapsack ← CreateKnapsack()
28:            knapsack.id ← kID + 1
29:            knapsack.partial ← ∅
30:        end if
31:        knapsack.data ← Append(subgraph)
32:        knapsack.subgraphs ← knapsack.subgraphs + 1
33:        iOffset ← knapsack.Index.LastOffset
34:        knapsack.UpdateIndex(subgraph.nID, iOffset + 1, iOffset + Size(chunk))
35:    end if
36: end function
```

도면5

**Algorithm 2** Knapsack Storage and Indexing

**Input:** Knapsack, Path
**Output:** Knapsack Stored with Index Record

```
1: function StoreKnapsack(knapsack, path)
2:     irecord ← new KnapsackIndexRecord()
3:     irecord.put("id", knapsack.id)
4:     irecord.put("subgraphs", knapsack.subgraphs)
5:     irecord.put("partial", knapsack.partial)
6:     irecord.put("path", Path)
7:     vertices ← ∅
8:     for each record ∈ knapsack.Index do
9:         vertices.put(record.id)
10:    end for
11:    irecord.put("vertices", vertices)
12:    CreateIndexEntry(irecord)
13:    StoreInHDFS(knapsack, path)
14: end function
```

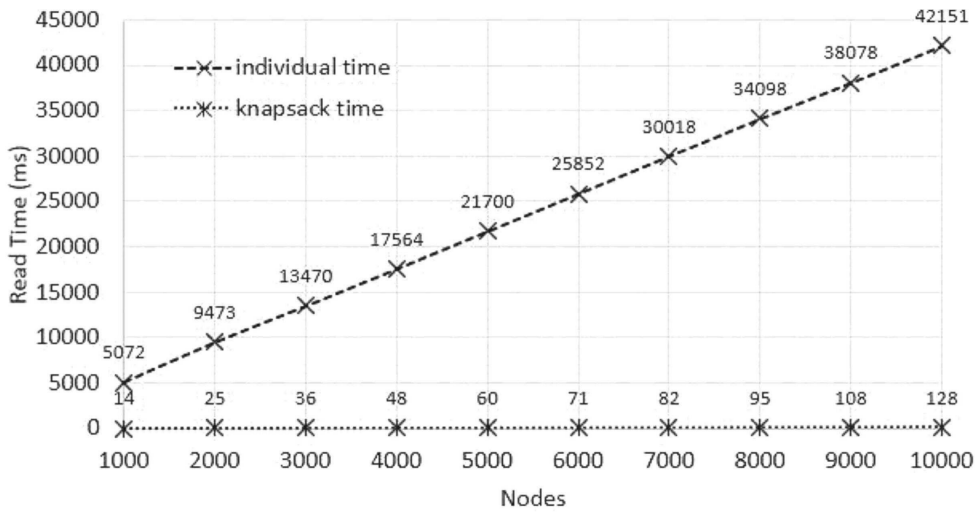
도면6

**Algorithm 3** Knapsack Retrieval for Training

**Input:** VertexID
**Output:** Knapsack With the given Vertex

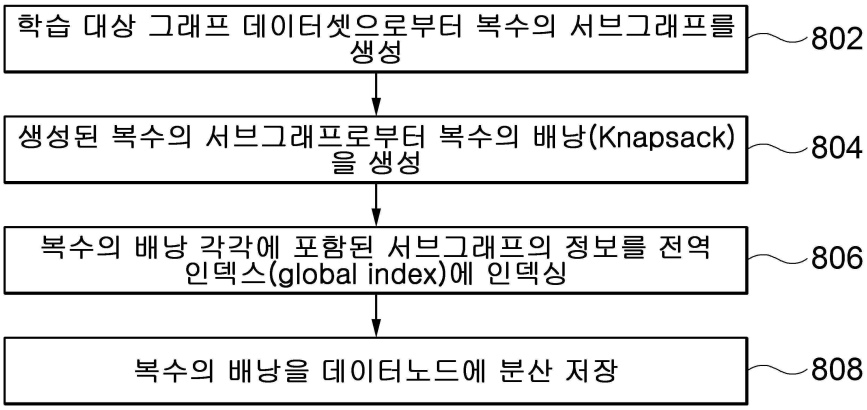
```
1:  function GetKnapsack(vID)
2:      knapsack = ∅
3:      irecord ← Lookup(vID)
4:      if irecord ≠ ∅ then
5:          path ← irecord.path
6:          knapsack ← ReadFromHDFS(path)
7:          if knapsack.partial ≠ ∅ then
8:              repeat
9:                  path ← Find(knapsack.partial)
10:                 chunk ← ReadFromHDFS(path)
11:                 Merge(knapsack, chunk)
12:             until knapsack.partial = ∅
13:         end if
14:     end if
15:     return knapsack
16: end function
```

도면7



도면8

800


학습 대상 그래프 데이터셋으로부터 복수의 서브그래프를 생성
802
생성된 복수의 서브그래프로부터 복수의 배낭(Knapsack)을 생성
804
복수의 배낭 각각에 포함된 서브그래프의 정보를 전역 인덱스(global index)에 인덱싱
806
복수의 배낭을 데이터노드에 분산 저장
808