



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2012-0084100
(43) 공개일자 2012년07월27일

(51) 국제특허분류(Int. Cl.)

G06F 15/16 (2006.01)

(21) 출원번호 10-2011-0005424

(22) 출원일자 2011년01월19일

심사청구일자 2011년01월19일

(71) 출원인

충남대학교산학협력단

대전광역시 유성구 대학로 99 (궁동, 충남대학교)

(72) 발명자

이영석

대전광역시 유성구 배울2로 3, 대덕테크노밸리 8
단지 금성백조아파트 809동 602호 (관평동)

이연희

대전광역시 유성구 노은로 353, 송림마을 3단지
304동 1701호 (하기동)

(74) 대리인

김원준

전체 청구항 수 : 총 5 항

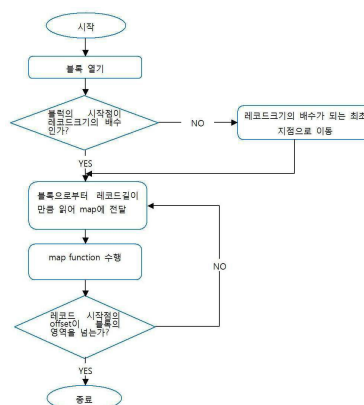
(54) 발명의 명칭 하둡 맵리듀스에서 바이너리 형태의 데이터 분석을 위한 입력포맷 및 이를 이용한 바이너리 데이터의 분석방법

(57) 요약

본 발명은 (A) 바이너리 데이터의 레코드의 길이를 입력받는 단계; (B) 하둡분산파일시스템(HDFS)에 저장된 데이터 블록 중 처리해야 될 데이터 블록에서 레코드의 길이의 n배수가 되는 지점 중 블록 시작점에 가장 가까운 값을 시작점으로 이전 InputSplit과 자신의 InputSplit의 경계를 설정하는 것에 의해 InputSplit을 정의하는 단계; (C) 상기에서 정의된 자신의 InputSplit 전체 영역에 대해 시작점으로부터 레코드의 길이만큼씩 읽는 일을 수행하는 RecordReader를 생성하고 이를 반환하는 단계; 및 (D) 상기 RecordReader를 통해 (Key, Value)를 (LongWritable, BytesWritable)의 형태로 레코드들을 추출하는 단계;를 포함하여 이루어지는 것을 특징으로 하는 고정길이의 레코드를 갖는 바이너리 데이터를 분산처리하기 위한 하둡 맵리듀스에서의 입력포맷과, 상기 입력 포맷을 이용한 바이너리 데이터의 분석방법에 관한 것이다.

본 발명의 입력포맷에 의하면, 고정길이의 바이너리 데이터를 하둡 환경에서 분산 처리할 때 데이터 포맷의 변환 작업 없이 처리가 가능하므로, 다른 형태의 데이터에 비해 적은 저장공간을 요하며 빠른 처리 속도를 가능하게 한다.

대표도 - 도2



이 발명을 지원한 국가연구개발사업

과제고유번호	2010-0263
부처명	정보통신연구진흥원
연구사업명	대학 IT 연구센터 육성지원사업
연구과제명	해킹?바이러스 대응기술 연구
주관기관	충남대학교
연구기간	2003.08.01 ~ 2011.08.31이 발명을 지원한 국가연구개발사업
과제고유번호	2010-0582
부처명	한국산업기술 평가관리원
연구사업명	IT 핵심 기술개발 사업
연구과제명	초정밀 측정 및 분석기술 연구
주관기관	충남대학교
연구기간	2008.03.01 ~ 2013.02.28

특허청구의 범위

청구항 1

하둡 맵리듀스에서의 입력포맷에 있어서,

- (A) 바이너리 데이터의 레코드의 길이를 입력받는 단계;
- (B) 하둡분산파일시스템(HDFS)에 저장된 데이터 블록 중 처리해야 될 데이터 블록에서 레코드의 길이의 n배수가 되는 지점 중 블록 시작점에 가장 가까운 값을 시작점으로 이전 InputSplit과 자신의 InputSplit의 경계를 설정하는 것에 의해 InputSplit을 정의하는 단계;
- (C) 상기에서 정의된 자신의 InputSplit 전체 영역에 대해 시작점으로부터 레코드의 길이만큼씩 읽는 임무를 수행하는 RecordReader를 생성하고 이를 반환하는 단계; 및
- (D) 상기 RecordReader를 통해 (Key, Value)를 (LongWritable, BytesWritable)의 형태로 레코드들을 추출하는 단계;

를 포함하여 이루어지는 것을 특징으로 하는 고정길이의 레코드를 갖는 바이너리 데이터를 분산처리하기 위한 하둡 맵리듀스에서의 입력포맷.

청구항 2

제 1 항에 있어서,

상기 바이너리 데이터는 NetFlow v5의 플로우 데이터인 것을 특징으로 하는 하둡 맵리듀스에서의 입력포맷.

청구항 3

제 2 항에 있어서,

상기 Key는 파일로부터의 offset 값, 출발지 포트번호, 도착지 포트번호, 출발지 IP 주소, 도착지 IP주소, 플로우의 타임스탬프와 출발지 포트번호를 하나의 바이트배열로 구성한 값, 플로우의 타임스탬프와 도착지 포트번호를 하나의 바이트배열로 구성한 값, 플로우의 타임스탬프와 출발지 IP 주소를 하나의 바이트배열로 구성한 값 및 플로우의 타임스탬프와 출발지 혹은 도착지 IP 주소를 하나의 바이트배열로 구성한 값으로 구성된 군으로부터 선택된 하나이며,

상기 Value는 패킷의 수, 바이트의 수 및 플로우의 수로 구성된 군으로부터 선택된 하나 이상을 하나의 바이트배열로 구성한 값인 것을 특징으로 하는 하둡 맵리듀스에서의 입력포맷.

청구항 4

제 1 항에 있어서,

상기 바이너리 데이터는 텍스트 형으로 변환되기 전의 원시 형태의 유전자 염기서열 데이터, 위치정보와 색상정보로 구성된 픽셀의 조합으로 구성된 비트맵 방식의 이미지 파일을 구성하는 데이터 또는 고정길이의 청크 사이즈를 갖는 웨이브 파일을 구성하는 데이터인 것을 특징으로 하는 하둡 맵리듀스에서의 입력포맷.

청구항 5

제 1 항 내지 제 4 항 중 어느 하나의 입력포맷을 이용하여 하둡 맵리듀스에서 고정길이의 레코드를 갖는 바이너리 데이터를 분석하는 방법에 관한 것으로,

- (A) 하둡분산파일시스템의 데이터블록으로부터 고정길이의 레코드를 갖는 바이너리 데이터를 읽어오는 단계;
 - (B) 제 1 항의 입력포맷에 의해 바이너리 형태의 데이터로부터 (Key, Value)를 생성하는 단계;
 - (C) 상기 (Key, Value)값을 사용하여 맵리듀스를 진행하는 단계;
 - (D) 상기 맵리듀스의 결과를 바이너리 데이터의 형태로 변환하는 단계; 및
 - (E) 상기 바이너리 형태로 변환된 데이터를 하둡분산파일시스템에 저장하는 단계;
- 를 포함하여 진행되는 것을 특징으로 하는 고정길이의 레코드를 갖는 바이너리 데이터를 분석하는 방법.

명세서

기술분야

[0001] 본 발명은 하둡 맵리듀스에서 고정된 길이의 레코드를 갖는 바이너리 포맷의 데이터를 처리하기 위한 새로운 입력포맷과, 상기 입력 포맷을 이용한 바이너리 데이터의 분석방법에 관한 것이다.

배경기술

[0002] 하둡(Hadoop)은 너치(Nutch)의 분산처리를 지원하기 위해 개발된 것으로, 수백 기가바이트~테라바이트 혹은 페타바이트 크기의 데이터를 처리할 수 있는 어플리케이션을 제작하고 운영할 수 있는 기반을 제공해 주는 데이터 처리 플랫폼이다. 하둡이 처리하는 데이터의 크기가 통상 최소 수백 기가바이트 수준이기 때문에 데이터는 하나의 컴퓨터에 저장되는 것이 아니라 여러 개의 블록으로 나누어져 여러 개의 컴퓨터에 분산 저장된다. 따라서 하둡은 입력되는 데이터를 나누어 처리할 수 있도록 하는 하둡 분산 파일 시스템(HDFS: Hadoop Distributed File System)을 포함하며, 분산 저장된 데이터들은 대용량 데이터를 클러스터 환경에서 병렬 처리하기 위해 개발된 MapReduce 과정에 의해 처리되어진다.

[0003] 도 1은 하둡 맵리듀스에서 잡(Job) 처리 시의 데이터의 흐름을 보여주는 개념도이다. 입력파일(input file)은 맵리듀스가 수행될 데이터가 저장된 것으로 통상은 HDFS에 저장되어 진다. 하둡은 텍스트 포맷의 데이터 뿐 아니라 다양한 형태의 데이터 포맷을 지원한다.

[0004] 클라이언트의 요청에 의해 Job이 시작되면, 입력포맷(InputFormat, 101)은 입력파일을 어떻게 나누고, 읽을 것인가를 결정하게 된다. 즉 해당 블록의 데이터에 대해 입력 파일을 나누어 InputSplit을 반환하는 한편, InputSplit을 맵퍼(mapper)가 읽을 수 있는 (key, value) 형태로 변환한 RecordReader(102)를 생성하여 반환한다. InputSplit는 맵리듀스에서 단일의 맵작업이 처리하는 데이터의 단위이다. 하둡에서는 TextInputFormat, KeyValueInputFormat, SequenceInputFormat과 같은 유형의 입력포맷이 있다. 대표적인 입력포맷은 TextInputFormat으로서 각 라인을 기준으로 블록단위로 저장된 입력파일을 나누어 논리적인 입력단위인 InputSplit을 구성하며, 이 InputSplit으로부터 (LongWritable, Text)의 형태의 레코드를 추출하는 임무를 수행하는 LineRecordReader를 반환한다.

[0005] 반환된 RecordReader는 통상적인 Map 과정 중에 InputSplit에서 키와 값의 쌍으로 구성된 레코드를 읽어 맵퍼에 넘겨주는 역할을 수행한다. 맵퍼는 이 레코드를 Map에 정의된 처리과정을 거치면서 새로운 키와 값으로 구성된 레코드로 생성한다. 출력포맷(OutputFormat, 103)은 MapReduce 과정에서 생성한 데이터를 HDFS에 파일로 출력하기 위한 포맷으로서, 출력포맷은 subclass인 RecordWriter(104)를 통하여 MapReduce 처리의 결과로 받은 키와 값의 쌍으로 구성된 레코드를 HDFS에 쓰는 것에 의해 데이터 처리 과정을 종료하게 된다.

[0006] Hadoop은 웹 크롤링의 특성에 맞게 텍스트 데이터의 처리를 위한 다양한 형태의 입력 포맷과 출력 포맷을 제공하며, 이 중 시퀀스파일포맷은 텍스트 이외의 데이터 포맷에 대한 입력과 출력을 제공한다. deflate, gzip, ZIP, bzip2, and LZ0 등의 압축파일의 입출력도 지원하며, 이러한 압축파일포맷은 저장공간의 효율을 높일 수 있다는 장점이 있다. 그러나, 압축파일포맷에 의해 입력파일을 처리하기 위해서는 MapReduce 작업이 시작되기 전에 압축을 해제하고 처리한 결과를 다시 압축하는 단계가 필요하므로 처리 속도가 낮아지는 문제가 있다. 시퀀스파일포맷은 바이너리를 포함한 다양한 포맷의 데이터를 담을 수 있는 틀을 제공하지만, 담고자 하는 원시 데이터를 일련의 시퀀스 형태로 변환해야만 하는 또 다른 변환과정이 요구된다.

[0007] 이와 같은 이유로 이미지, 통신 패킷 등 바이너리 형태를 띠고 있는 대량의 데이터를 하둡 분산환경에서 처리하기 위해서는 텍스트로의 변환이나 다른 하둡에서 인식할 수 있는 형태로의 데이터 변환이 필요하다. 이러한 변환 작업은 단일 시스템에 의해 변환하고자 하는 파일을 읽고 변환하여 저장을 위해 다시 쓰는 과정으로 이루어지며, 하둡의 분산시스템을 이용하여 처리 성능을 향상시키려는 근본적인 취지에 어긋나는 것이다. 이에, 바이너리 데이터를 하둡 분산환경에서 처리하기 위해서는 보다 효과적인 방법의 개발이 필요하다.

발명의 내용

해결하려는 과제

[0008] 상기와 같은 종래기술의 문제점을 해소하기 위한 본 발명의 목적은 NetFlow v5와 같이 고정된 길이의 데이터 레코드 블록을 갖는 바이너리 데이터를 하둡 분산처리 시스템에서 효과적으로 처리할 수 있도록 하는 입력포맷을 제공하는 것이다.

[0009] 본 발명의 또 다른 목적은 상기의 입력포맷을 이용하여 하둡 맵리듀스에 의해 고정된 길이의 레코드를 갖는 바이너리 데이터를 처리하는 방법을 제공하는 것이다.

과제의 해결 수단

[0010] 진술한 목적을 달성하기 위한 본 발명은 고정길이의 레코드를 갖는 바이너리 데이터를 분산처리하기 위한 하둡 맵리듀스에서의 입력포맷에 관한 것으로, (A) 바이너리 데이터의 레코드의 길이를 입력받는 단계; (B) 하둡분산 파일시스템(HDFS)에 저장된 데이터 블록 중 처리해야 될 데이터 블록에서 레코드의 길이의 n배수가 되는 지점 중 블록 시작점에 가장 가까운 값을 시작점으로 이전 InputSplit과 자신의 InputSplit의 경계를 설정하는 것에 의해 InputSplit을 정의하는 단계; (C) 상기에서 정의된 자신의 InputSplit 전체 영역에 대해 시작점으로부터 레코드의 길이만큼씩 읽는 일을 수행하는 RecordReader를 생성하고 이를 반환하는 단계; 및 (D) 상기 RecordReader를 통해 (Key, Value)를 (LongWritable, BytesWritable)의 형태로 레코드들을 추출하는 단계;를 포함하여 이루어지는 것을 특징으로 한다.

[0011] 또한 본 발명은 상기 입력포맷을 이용하여 하둡 맵리듀스에서 고정길이의 레코드를 갖는 바이너리 데이터를 분석하는 방법에 관한 것이다. 보다 상세하게는, (A) 하둡분산파일시스템의 데이터블록으로부터 고정길이의 레코드를 갖는 바이너리 데이터를 읽어오는 단계; (B) 제 1 항의 입력포맷에 의해 바이너리 형태의 데이터로부터 (K, V)를 생성하는 단계; (C) 상기 (K, V)값을 사용하여 맵리듀스를 진행하는 단계; (D) 상기 맵리듀스의 결과를 바이너리 데이터의 형태로 변환하는 단계; 및 (E) 상기 바이너리 형태로 변환된 데이터를 하둡분산파일시스템에 저장하는 단계;를 포함하여 진행되는 것을 특징으로 하는 고정길이의 레코드를 갖는 바이너리 데이터를 분석하는 방법에 관한 것이다.

발명의 효과

[0012] 이상과 같이 본 발명의 입력포맷에 의하면, NetFlow v5와 같은 고정길이의 바이너리 데이터를 하둡 환경에서 분산 처리할 때 데이터 포맷의 변환작업 없이 처리가 가능하므로, 다른 형태의 데이터에 비해 적은 저장공간을 요하며 빠른 처리 속도를 가능하게 한다.

[0013] 본 발명의 바이너리 형태의 데이터 분석 방법은 하둡 시스템을 이용한 패킷의 패턴매칭과 같은 응용을 통한 침입탐지시스템의 구성이나, 이미지 데이터, 유전자 정보, 암호처리 등과 같은 바이너리 데이터를 다루는 분야의 분석에 활용될 수 있다.

도면의 간단한 설명

[0014] 도 1은 하둡에서 잡(Job) 처리 시의 데이터의 흐름을 보여주는 개념도.

도 2는 하둡의 각 클러스터 노드들이 데이터 블록을 읽어 처리하는 절차를 보여주는 순서도.

도 3은 본 발명의 입력과 출력 인터페이스를 하둡의 API 구조에 대비하여 보여주는 클래스 계층 구조도.

도 4는 본 발명의 방법과 종래의 TextInputFormat을 이용한 방법에 의해 NetFlow v5 플로우를 처리하는 과정을 비교하여 보여주는 비교도.

발명을 실시하기 위한 구체적인 내용

- [0015] 이하 첨부된 도면을 참조하여 본 발명을 보다 상세히 설명한다. 그러나 이러한 도면은 본 발명의 기술적 사상의 내용과 범위를 쉽게 설명하기 위한 예시일 뿐, 이에 의해 본 발명의 기술적 범위가 한정되거나 변경되는 것은 아니다. 또한 이러한 예시에 기초하여 본 발명의 기술적 사상의 범위 안에서 다양한 변형과 변경이 가능함은 당업자에게는 당연할 것이다.
- [0016] 본 발명은 고정길이의 레코드를 갖는 바이너리 데이터를 분산처리하기 위한 하둡 맵리듀스에서의 입력포맷에 관한 것으로, (A) 바이너리 데이터의 레코드의 길이를 입력받는 단계; (B) 하둡분산파일시스템(HDFS)에 저장된 데이터 블록 중 처리해야 될 데이터 블록에서 레코드의 길이의 n배수가 되는 지점 중 블록 시작점에 가장 가까운 값을 시작점으로 이전 InputSplit과 자신의 InputSplit의 경계를 설정하는 것에 의해 InputSplit을 정의하는 단계; (C) 상기에서 정의된 자신의 InputSplit 전체 영역에 대해 시작점으로부터 레코드의 길이만큼씩 읽는 일을 수행하는 RecordReader를 생성하고 이를 반환하는 단계; 및 (D) 상기 RecordReader를 통해 (Key, Value)를 (LongWritable, BytesWritable)의 형태로 레코드들을 추출하는 단계;를 포함하여 이루어지는 것을 특징으로 하는 고정길이의 레코드를 갖는 바이너리 데이터를 분산처리하기 위한 하둡 맵리듀스에서의 입력포맷에 관한 것이다.
- [0017] 상기 고정길이의 레코드를 갖는 바이너리 데이터로는 NetFlow v5의 플로우 데이터, 텍스트 형으로 변환되기 전의 원시 형태의 유전자 염기서열 데이터, 위치정보와 색상정보로 구성된 픽셀의 조합으로 구성된 비트맵 방식의 이미지 파일을 구성하는 데이터, 고정길이의 청크 사이즈를 갖는 웨이브 파일을 구성하는 데이터를 예로 들 수 있으나, 이에 한정되는 것은 아니며 레코드의 길이가 고정된 바이너리 데이터에 대해서는 모두 적용이 가능하다.
- [0018] 도 2는 본 발명의 입력포맷에 의해 하둡의 맵리듀스 과정을 진행하기 위해 각 클러스터 노드들이 데이터 블록을 읽어 처리하는 절차를 보여주는 순서도이다.
- [0019] 먼저 JobClient를 통해 바이너리 데이터의 레코드의 길이를 입력받는다. 이 때 값을 입력받는 방법으로는 Configuration Property를 이용하여 특정 Property에 레코드 크기에 대한 정보를 할당하여 모든 클러스터 노드들이 공유하도록 할 수 있다. 데이터 처리를 위해 블록을 열면, 블록의 시작점이 레코드의 길이의 n배수가 되는 지점인지 확인한다. 이때, n은 0 또는 자연수이다. 블록의 시작점이 레코드 길이의 n배수가 되는 지점이라면, 그 지점을 InputSplit의 시작점으로 정의한다. 블록의 시작점이 레코드 길이의 n배수가 되는 지점이 아니라면, 1byte씩 이동하면서 다시 레코드 길이의 n배수가 되는 지점인지 확인하는 과정을 거쳐 레코드 길이의 n배수가 되는 최초 지점으로 이동하여 InputSplit의 시작점으로 정의한다. 즉, 데이터 블록의 InputSplit은 레코드 길이의 n배수가 되는 지점 중 블록 시작점에 가장 가까운 값으로부터 다음 데이터 블록에 대한 InputSplit 시작점 전까지를 데이터 블록에 대한 InputSplit로 정의한다.
- [0020] InputSplit가 정의되면 해당 InputSplit으로부터 맵작업을 수행하기 위하여 InputSplit의 시작점으로부터 레코드의 길이만큼씩 읽어 레코드를 추출하는 임무를 수행하는 RecordReader를 생성하여 반환한다. 이 때 RecordReader에 의해 Map으로 전달되는 (Key, Value)는 하둡의 (LongWritable, BytesWritable) 형의 Writable 클래스 타입이며, 예를 들면 (파일 시작점으로부터의 offset 값, 레코드 데이터)의 형태로 레코드들을 추출하여 맵에 전달한다.
- [0021] NetFlow v5의 플로우 데이터를 예로 들어 설명하면, NetFlow v5 패킷 데이터를 Value로 기재할 수 있다. 즉, 상기 Value는 패킷의 수, 바이트의 수 및 플로우의 수로 구성된 군으로부터 선택된 하나 이상을 하나의 바이트 배열로 구성한 값일 수 있다.
- [0022] Key 역시 처리하고자 하는 데이터와 Job의 성격에 따라 offset 값이 아닌 레코드의 인덱스로서의 의미가 있는 다른 값을 설정할 수도 있다. 즉, NetFlow 분석에서 포트번호 별 전체 패킷의 수, 바이트의 수, 플로우의 수

를 구하고자하는 경우, 파일로부터의 offset 값이 아닌 포트번호를 Key로 사용할 수 있으며, 출발지 IP에 따른 전체 패킷의 수, 바이트의 수, 플로우의 수를 구하고자하는 경우, 출발지 IP를 Key로 정의한다. 일정 시간간격에 따른 포트번호 별 전체 패킷의 수, 바이트의 수, 플로우의 수를 구하고자하는 경우라면, 플로우의 타임스탬프와 포트번호를 하나의 바이트배열로 구성하여 Key키로 전달한다. 일정 시간간격에 따른 출발지 IP별 플로우 데이터를 분석하고자 한다면, 플로우의 타임스탬프와 출발지 IP를 하나의 바이트배열로 구성하여 Key키로 전달하는 것과 같이 패킷을 구성하는 모든 항목에 대한 모든 조합을 키로 구성하여 맵리듀스에 의해 분석하는 것이 가능하다. 이와 같이, 상기 Key는 파일로부터의 offset 값, 출발지 포트번호, 도착지 포트번호, 출발지 IP 주소, 도착지 IP주소, 플로우의 타임스탬프와 출발지 포트번호를 하나의 바이트배열로 구성한 값, 플로우의 타임스탬프와 도착지 포트번호를 하나의 바이트배열로 구성한 값, 플로우의 타임스탬프와 출발지 IP 주소를 하나의 바이트배열로 구성한 값 및 플로우의 타임스탬프와 출발지 혹은 도착지 IP 주소를 하나의 바이트배열로 구성한 값으로 구성된 군으로부터 선택된 하나를 사용할 수 있다.

[0023] 상기와 같은 방법에 의해 블록 중 레코드의 최초 시작점을 이전 InputSplit과의 경계로 하는 InputSplit을 정의하고 RecordReader를 반환하면, Mapper는 RecordReader를 이용하여 하나의 레코드 씩 InputSplit으로부터 읽어 Map Function을 수행한다. 이 때 RecordReader는 전달하고자 하는 레코드의 파일로부터의 offset이 블록의 InputSplit의 모든 레코드들이 처리되었는지 판별하기 위해 전달하고자 하는 레코드 시작점의 offset이 자신이 처리할 블록의 영역을 벗어나는 지 확인함으로써 이어지는 블록의 InputSplit의 영역을 침범하지 않도록 한다. 만일, offset이 블록의 영역을 넘지 않는다면 RecordReader는 offset이 블록의 영역을 넘게 될 때까지 레코드를 읽어 생성하는 작업을 반복한다.

[0024] 이와 같은 본 발명의 RecordReader를 포함하는 입력포맷을 BinaryInputFormat이라 명명한다.

[0025] 도 3은 본 발명에서 구현한 입력포맷을 종래의 하둡의 응용프로그래밍인터페이스(API, Application Programming Interface)의 구조에 대비하여 보여주는 클래스 계층 구조도이다. 도 3의 클래스의 명칭과 계층은 구현 방법에 따라 다소 변경하여 구현할 수 있다.

[0026] 본 발명은 또한 상기 BinaryInputFormat을 이용하여 하둡 맵리듀스에서 고정길이의 레코드를 갖는 바이너리 데이터를 분석하는 방법에 관한 것으로, 보다 상세하게는, (A) 하둡분산파일시스템의 데이터블록으로부터 고정길이의 레코드를 갖는 바이너리 데이터를 읽어오는 단계; (B) 본 발명의 BinaryInputFormat에 의해 바이너리 형태의 데이터로부터 (K, V)를 생성하는 단계; (C) 상기 (K, V)값을 사용하여 맵리듀스를 진행하는 단계; (D) 상기 맵리듀스의 결과를 바이너리 데이터의 형태로 변환하는 단계; 및 (E) 상기 바이너리 형태로 변환된 데이터를 하둡분산파일시스템에 저장하는 단계를 포함하여 진행되는 것을 특징으로 하는 고정길이의 레코드를 갖는 바이너리 데이터를 분석하는 방법에 관한 것이다.

[0027] 도 4는 종래기술에 의한 TextInputFormat을 이용하여 NetFlow v5 플로우를 처리하는 과정(400)과, 본 발명에 의한 BinaryInputFormat을 이용하여 NetFlow v5 플로우를 처리하는 과정(410)을 비교하여 보여주는 도면이다.

[0028] 도 4를 참조하여 TextInputFormat을 이용하여 NetFlow v5 플로우를 처리하는 과정을 설명한다. 먼저 nProbe와 같은 플로우 송신부(Flow Exporter, 401)는 패킷을 수집하여 실시간으로 플로우 정보를 생성하고 플로우 수집기(Flow Collector, 402)로 전송한다. 플로우 수집기(402)는 전송받은 플로우를 캡처하여 플로우 데이터를 저장한다. 저장된 플로우 데이터는 바이너리 포맷이므로 flow-print와 같은 텍스트 변환기(Text Converter, 403)를 실행하여 저장된 플로우 데이터를 읽고 이를 하둡에서 처리할 수 있는 텍스트 형태의 플로우 데이터로 변환한 후 이를 HDFS에 저장한다. flow-print 명령 과정에서 플로우 패킷의 헤더는 자동으로 제외된다.

[0029] 일단 하둡 맵리듀스를 이용한 플로우 분석기(Flow Analyzer, 404)의 플로우 분석이 실행되면 저장된 HDFS로부터 블록단위의 플로우를 읽어와 TextInputFormat(405)을 이용하여 텍스트 형태의 데이터로부터 (K, V)를 생성한다. 생성된 (K, V) 값을 이용하여 맵, 리듀스 작업을 거친 결과물은 다시 TextOutputFormat(406)에 의해 텍스트 형태의 결과로 HDFS에 저장된다.

[0030] 이에 반해 본 발명의 방법에 의하면 플로우 송신부(Flow Exporter, 411)에 의해 수집된 패킷으로부터 생성된 플로우 정보가 플로우 수집기(Flow Collector, 412)로 전송되면, 플로우 수집기(412)는 전송받은 플로우를 캡처하

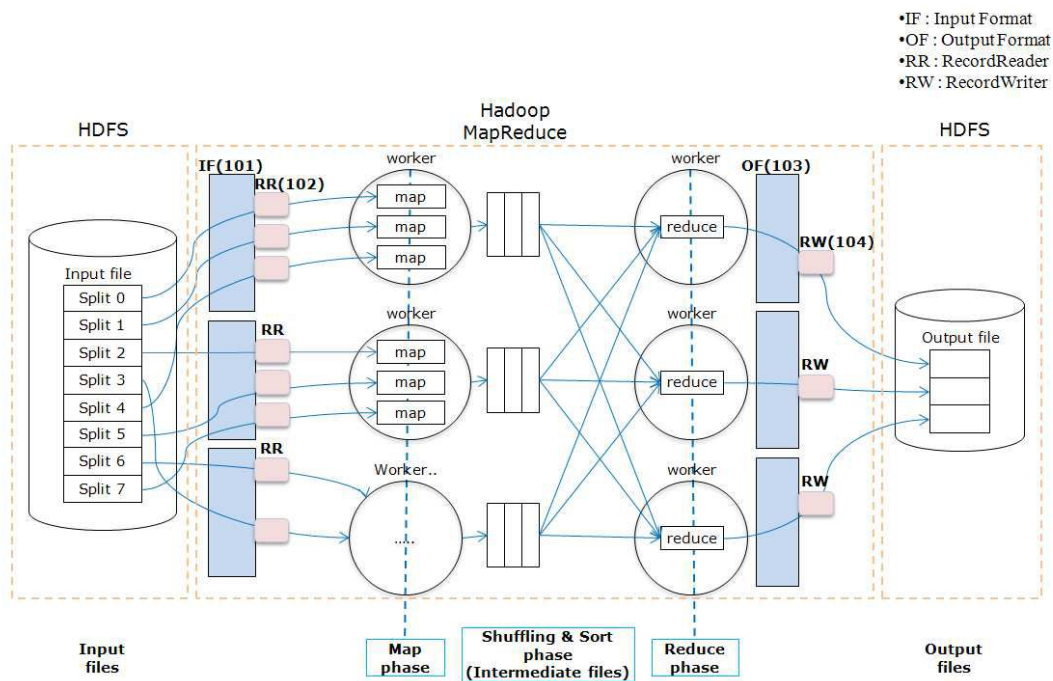
여 플로우 패킷의 헤더를 제거한 후 별도의 변환 과정이 없이 이를 HDFS에 저장한다. BinaryInputFormat을 이용한 하둡 맵리듀스 플로우 분석기(413)의 플로우 분석이 실행되면 저장된 HDFS로부터 블록단위의 플로우를 읽어와 BinaryInputFormat(414)을 이용하여 블록으로부터 바이너리 형태의 레코드를 추출하여 Map에 전달한다. 전달된 레코드들은 Map과 Reduce 과정을 거친 후, 그 결과를 바이너리 형태로 출력하여 HDFS에 저장한다.

[0031] 맵리듀스의 결과를 특정 형태로 출력하는 것은 출력포맷(OutputFormat)에 의해 이루어진다. 본 발명에서의 출력포맷은 HDFS로의 파일 출력을 위한 클래스인 FileOutputFormat을 확장하여 간단히 구현할 수 있으며 이를 BinaryOutputFormat으로 명명한다. 출력 레코드의 Key와 Value를 모두 BytesWritable로 하여 MapReduce의 분석결과로써의 BytesWritable 형으로 된 바이너리 데이터를 담아 HDFS에 출력할 수 있다.

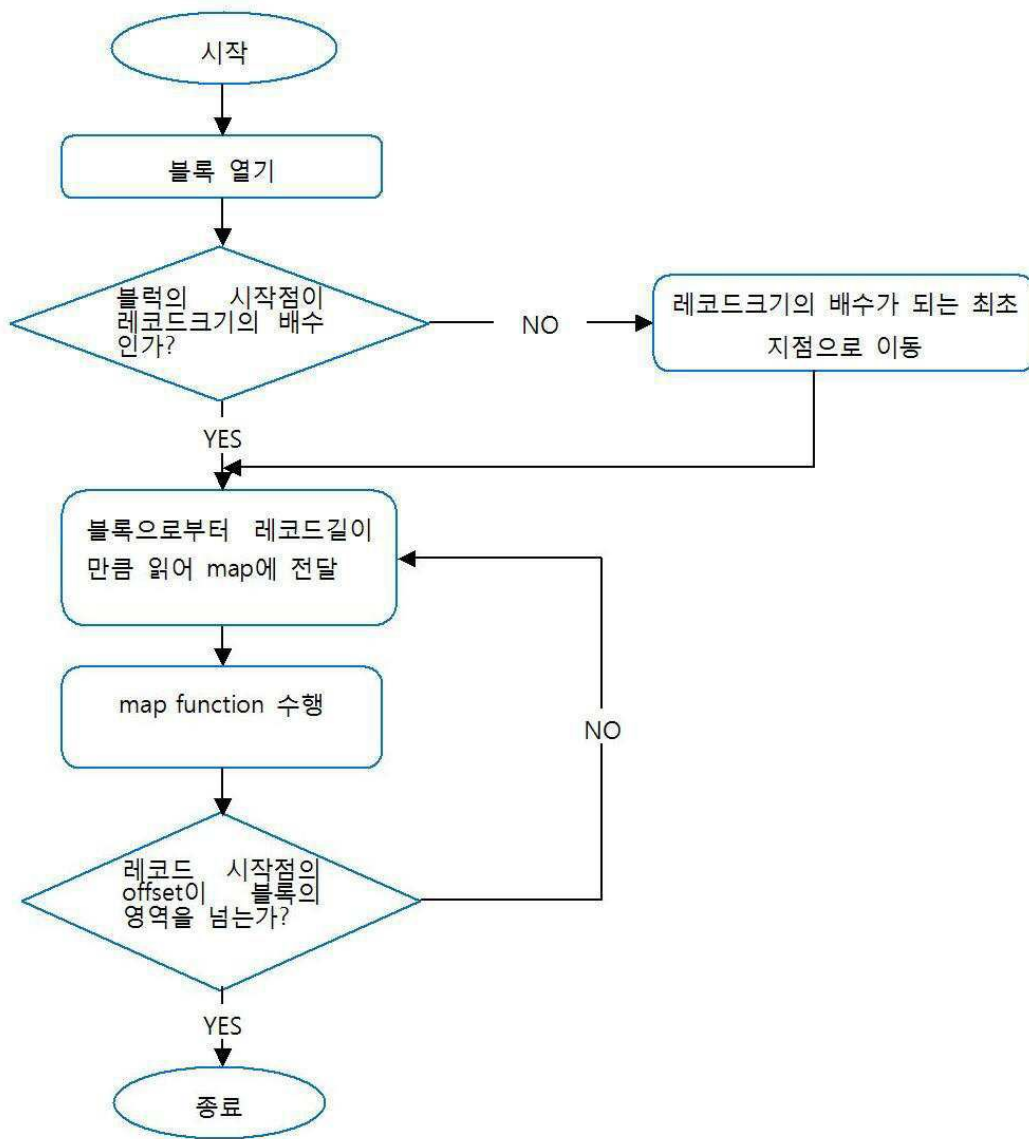
[0032] 상기와 같이 본 발명의 BinaryInputFormat을 사용하면 별도의 데이터 변환의 절차가 필요없으므로 시간과 저장 공간을 절약하여 데이터 처리 효율을 높일 수 있다.

도면

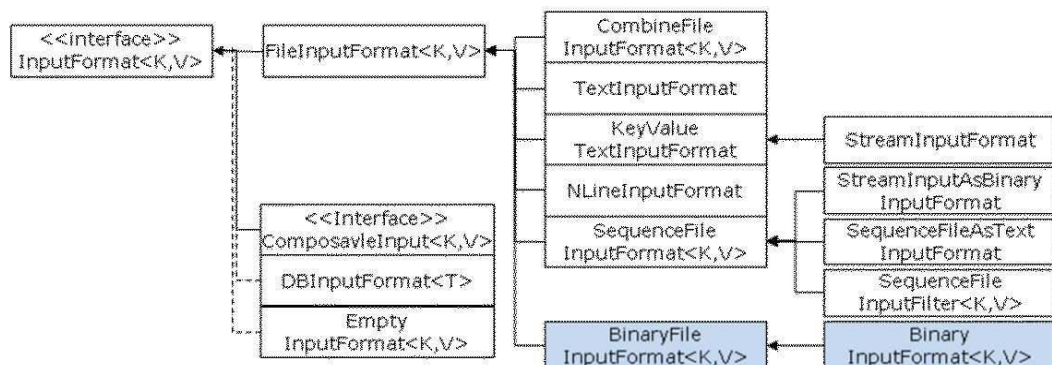
도면1



도면2



도면3



도면4

