



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2019-0109638
(43) 공개일자 2019년09월26일

(51) 국제특허분류(Int. Cl.)

G06F 16/00 (2019.01)

(52) CPC특허분류

G06F 16/17 (2019.01)

G06F 16/182 (2019.01)

(21) 출원번호 10-2018-0025780

(22) 출원일자 2018년03월05일

심사청구일자 2018년03월05일

(71) 출원인

울산과학기술원

울산광역시 울주군 언양읍 유니스트길 50

(72) 발명자

최영리

울산광역시 울주군 언양읍 유니스트길 50 울산과학기술대학교

황은지

울산광역시 울주군 언양읍 유니스트길 50 울산과학기술대학교

김현구

울산광역시 울주군 언양읍 유니스트길 50 울산과학기술대학교

(74) 대리인

제일특허법인(유)

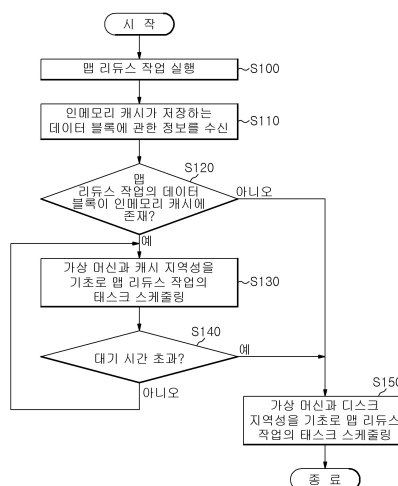
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼의 태스크 스케줄링 방법, 이를 위한 컴퓨터 프로그램 및 컴퓨터 판독 가능 기록 매체

(57) 요약

본 발명은 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법에 관한 것으로, 네임 노드와 데이터 노드를 포함하는 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법에 있어서, 맵 리듀스 작업(Map Reduce job)이 실행되면 상기 데이터 노드의 인메모리 캐시(in memory cache)가 저장하는 데이터 블록에 관한 정보를 상기 네임 노드로부터 수신하는 단계; 상기 데이터 블록에 관한 정보를 기초로 상기 맵 리듀스 작업에 이용되는 데이터 블록이 상기 인메모리 캐시에 존재하는지를 판단하는 단계; 및 상기 맵 리듀스 작업에 이용되는 데이터 블록이 상기 인메모리 캐시에 존재하면 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는 단계를 포함할 수 있다.

대표도 - 도3



명세서

청구범위

청구항 1

네임 노드와 데이터 노드를 포함하는 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법에 있어서,

맵 리듀스 작업(Map Reduce job)이 실행되면 상기 데이터 노드의 인메모리 캐시(in memory cache)가 저장하는 데이터 블록에 관한 정보를 상기 네임 노드로부터 수신하는 단계;

상기 데이터 블록에 관한 정보를 기초로 상기 맵 리듀스 작업에 이용되는 데이터 블록이 상기 인메모리 캐시에 존재하는지를 판단하는 단계; 및

상기 맵 리듀스 작업에 이용되는 데이터 블록이 상기 인메모리 캐시에 존재하면 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는 단계를 포함하는

분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법.

청구항 2

제 1 항에 있어서,

상기 스케줄링하는 단계는,

상기 맵 리듀스 작업의 태스크가 실행되는 복수의 물리 머신(Physical Machine) 내의 가상 머신에 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는

분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법.

청구항 3

제 2 항에 있어서,

상기 스케줄링하는 단계는,

상기 복수의 물리 머신 내에 각 물리 머신 당 복수의 가상 머신이 상주(co-resident)할 경우, 상기 맵 리듀스 작업의 태스크가 실행되는 가상 머신이 상주하는 물리 머신과 다른 물리 머신 내의 가상 머신에 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는

분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법.

청구항 4

제 1 항에 있어서,

상기 맵 리듀스 작업의 데이터 블록이 상기 인메모리 캐시에 존재하지 않거나 상기 스케줄링하는 단계의 대기 시간이 기 설정된 시간을 초과하면, 상기 가상 머신의 디스크에 저장되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는 단계를 더 포함하는

분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법.

청구항 5

제 1 항에 있어서,

상기 스케줄링 하는 단계는,

상기 빅데이터 처리 플랫폼의 서로 다른 물리 머신 내의 가상 머신에 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는

분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법.

청구항 6

네임 노드와 데이터 노드를 포함하는 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법에 있어서,

맵 리듀스 작업이 실행되면 상기 데이터 노드의 인메모리 캐시가 저장하는 데이터 블록에 관한 정보를 상기 네임 노드로부터 수신하는 단계;

상기 데이터 블록에 관한 정보를 기초로 상기 맵 리듀스 작업의 데이터 블록이 상기 인메모리 캐시에 존재하는지를 판단하는 단계; 및

상기 맵 리듀스 작업의 데이터 블록이 상기 인메모리 캐시에 존재하면 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는 단계를 수행하는 명령어를 포함하는 프로그램이 기록된

컴퓨터 판독 가능 기록 매체.

청구항 7

네임 노드와 데이터 노드를 포함하는 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법에 있어서,

맵 리듀스 작업이 실행되면 상기 데이터 노드의 인메모리 캐시가 저장하는 데이터 블록에 관한 정보를 상기 네임 노드로부터 수신하는 단계;

상기 데이터 블록에 관한 정보를 기초로 상기 맵 리듀스 작업의 데이터 블록이 상기 인메모리 캐시에 존재하는지를 판단하는 단계; 및

상기 맵 리듀스 작업의 데이터 블록이 상기 인메모리 캐시에 존재하면 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는 단계를 수행하는

컴퓨터 판독 가능 기록 매체에 저장된 컴퓨터 프로그램.

발명의 설명

기술 분야

[0001] 본 발명은 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼의 태스크 스케줄링 기술에 관한 것이다.

배경 기술

[0002] 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼 중 하나인 하둡(hadoop) 2.0의 Yarn에서 맵 리듀스 작업(Map Reduce job)은 적어도 하나의 태스크(task)로 구성된다. 그리고, Yarn의 스케줄러는 이러한 태스크를 노드 매니저(node manager)가 실행되는 계산 노드에서 스케줄링한다. 태스크가 계산 노드에서 스케줄링 되면 이러한 계산 노드에는 컨테이너(container)가 생성된다. 그리고, 태스크는 이와 같이 생성된 컨테이너에서 실행된다. 이 때, 태스크가 실행되는 계산 노드가 이러한 태스크가 처리해야 하는 데이터 블록을 저장하는 데이터 노드와 동일한 경우, 이러한 태스크는 데이터 로컬 태스크라고 지칭된다.

[0003] 최근 들어, 하둡 분산 파일 시스템(Hadoop Distribution File System, HDFS)에 인메모리 캐시(in-memory cache) 기능이 추가 되었다. 태스크를 실행함에 있어서 전송한 데이터 로컬 태스크의 경우 말고도, 하둡 분산 파일 인메모리 캐시 시스템의 인메모리 캐시로부터 그에 저장된 데이터 블록을 읽어들이어서 태스크를 실행할 수도 있는데, 이러한 태스크는 캐시 로컬 태스크라고 지칭된다. 이때, 인메모리 캐시에 이와 같이 데이터 블록이 저장되어 있음을 최대한 활용하면서 하둡 분산 파일 인메모리 캐시 시스템의 효율성을 기존보다 향상시키는 방안이 요구된다.

선행기술문헌

특허문헌

[0004] (특허문헌 0001) 한국등록특허 제10-1661475호, 2016.09.26 등록

발명의 내용

해결하려는 과제

[0005] 본 발명의 실시예에서는 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼, 특히 인메모리 캐시 기반의 하둡 분산 파일 시스템(HDFS)을 포함하는 빅데이터 처리 플랫폼에서의 가상화 환경(Virtualization Environment)과 캐시 지역성(Cache Locality)을 함께 고려한 태스크 스케줄링 기술을 제안하고자 한다.

[0006] 본 발명이 해결하고자 하는 과제는 상기에서 언급한 것으로 제한되지 않으며, 언급되지 않은 또 다른 해결하고자 하는 과제는 아래의 기재들로부터 본 발명이 속하는 통상의 지식을 가진 자에 의해 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0007] 본 발명의 실시예에 따르면, 네임 노드와 데이터 노드를 포함하는 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법에 있어서, 맵 리듀스 작업(Map Reduce job)이 실행되면 상기 데이터 노드의 인메모리 캐시(in memory cache)가 저장하는 데이터 블록에 관한 정보를 상기 네임 노드로부터 수신하는 단계; 상기 데이터 블록에 관한 정보를 기초로 상기 맵 리듀스 작업에 이용되는 데이터 블록이 상기 인메모리 캐시에 존재하는지를 판단하는 단계; 및 상기 맵 리듀스 작업에 이용되는 데이터 블록이 상기 인메모리 캐시에 존재하면 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는 단계를 포함하는 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 방법을 제공할 수 있다.

[0008] 여기서, 상기 스케줄링하는 단계는, 상기 맵 리듀스 작업의 태스크가 실행되는 복수의 물리 머신(Physical Machine) 내의 가상 머신에 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링할 수 있다.

[0009] 또한, 상기 스케줄링하는 단계는, 상기 복수의 물리 머신 내에 각 물리 머신 당 복수의 가상 머신이 상주(co-resident)할 경우, 상기 맵 리듀스 작업의 태스크가 실행되는 가상 머신이 상주하는 물리 머신과 다른 물리 머신 내의 가상 머신에 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링할 수 있다.

[0010] 또한, 상기 맵 리듀스 작업의 데이터 블록이 상기 인메모리 캐시에 존재하지 않거나 상기 스케줄링하는 단계의 대기 시간이 기 설정된 시간을 초과하면, 상기 가상 머신의 디스크에 저장되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링하는 단계를 더 포함할 수 있다.

[0011] 또한, 상기 스케줄링 하는 단계는, 상기 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼에서의 서로 다른 물리 머신 내의 가상 머신에 캐시되어 있는 데이터 블록을 활용하여 상기 맵 리듀스 작업의 태스크를 스케줄링할 수 있다.

발명의 효과

[0012] 본 발명의 실시예에 의하면, 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼, 특히 인메모리 캐시 기반의 하둡 분산 파일 시스템(HDFS)을 포함하는 빅데이터 처리 플랫폼에서의 가상화 환경과 캐시 지역성을 함께 고려하여 맵 리듀스 작업의 태스크를 스케줄링함으로써, 맵 리듀스 작업이 캐시된 입력 데이터를 효율적으로 사용할 수 있다.

도면의 간단한 설명

[0013] 도 1은 본 발명의 실시예에 따른 태스크 스케줄링을 위한 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼, 특히 인메모리 캐시 기반의 하둡 분산 파일 시스템과 빅데이터 처리 플랫폼의 시스템 블록도이다.

도 2는 도 1의 인메모리 캐시 기반의 하둡 분산 파일 시스템을 포함한 빅데이터 처리 플랫폼에서 수행되는 가상화 환경을 고려한 태스크 스케줄링을 예시적으로 설명하는 도면이다.

도 3은 본 발명의 실시예에 따른 분산 파일 시스템, 예컨대 인메모리 캐시 기반의 하둡 분산 파일 시스템을 포

합하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 과정을 예시적으로 설명하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

- [0014] 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 후술되어 있는 실시예들을 참조하면 명확해질 것이다. 그러나 본 발명은 이하에서 개시되는 실시예들에 한정되는 것이 아니라 다양한 형태로 구현될 수 있으며, 단지 본 실시예들은 본 발명의 개시가 완전하도록 하고, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것이며, 본 발명의 범주는 청구항에 의해 정의될 뿐이다.
- [0015] 본 발명의 실시예들을 설명함에 있어서 공지 기능 또는 구성에 대한 구체적인 설명은 본 발명의 실시예들을 설명함에 있어 실제로 필요한 경우 외에는 생략될 것이다. 그리고 후술되는 용어들은 본 발명의 실시예에서의 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.
- [0016] 본 발명의 실시예는 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼, 특히 인메모리 캐시 기반의 하둡 분산 파일 시스템(Hadoop Distribution File System, 이하 HDFS라 함)을 포함하는 빅데이터 처리 플랫폼에서의 가상화 환경과 캐시 지역성을 함께 고려하여 맵 리듀스 작업의 태스크를 스케줄링함으로써, HDFS의 맵 리듀스 작업이 캐시된 입력 데이터를 효율적으로 사용할 수 있는 기술을 제안하고자 한다.
- [0017] 이하, 첨부된 도면을 참조하여 본 발명의 실시예에 대해 상세히 설명하기로 한다.
- [0018] 도 1은 본 발명의 실시예에 따른 태스크 스케줄링을 위한 분산 파일 시스템을 포함하는 빅데이터 처리 플랫폼, 예컨대 인메모리 캐시 기반의 HDFS(10)를 포함하는 빅데이터 처리 플랫폼의 시스템 블록도이다.
- [0019] 도 1을 참조하면, 인메모리 캐시 기반의 HDFS(10)는 네임 노드(300)와 데이터 노드(400)를 포함할 수 있다. 인메모리 캐시 기반의 HDFS(10)를 포함하는 빅데이터 처리 플랫폼 중 하나인 하둡에서 애플리케이션이 실행되면, 생성된 컨테이너(500)에서 태스크가 실행된다. 아울러, 계산 노드의 컨테이너에 태스크를 스케줄링하기 위한 스케줄링 장치(100) 또한 빅데이터 처리 플랫폼에 포함된다. 여기서, 전술한 계산 노드, 데이터 노드(400) 및 인메모리 캐시(420)는 복수의 서버(컴퓨터)로 구성된 클러스터에서 실행 가능하며, 이러한 서버는 이하에서 노드로 지칭될 수 있다. 경우에 따라 노드는 계산 노드와 데이터 노드(400) 복수의 기능을 수행할 수 있으며, 계산 노드와 데이터 노드(400) 중 어느 하나의 기능만을 수행할 수도 있다.
- [0020] 데이터 노드(400)는 복수 개로 구성될 수 있으며, 파일이 블록 단위로 나뉜 결과물인 데이터 블록을 스토리지의 일종인 로컬 디스크(410)에 저장한다.
- [0021] 인메모리 캐시(420)는 인메모리 캐시 기반 HDFS(10)를 구성하는 복수의 서버에 포함된 메모리, 예를 들어 RAM(Random Access Memory)의 일정 부분에 형성될 수 있다.
- [0022] 데이터 노드(400)는 자신의 인메모리 캐시(420)에 어떤 데이터 블록이 저장되어 있는지를 주기적으로 네임 노드(300)에 보고(reporting)한다. 예를 들어, 데이터 노드(400)는 3초 단위의 하트비트(heartbeat)를 네임 노드(300)에 주기적으로 전송할 수 있다.
- [0023] 네임 노드(300)는 파일이나 데이터 블록의 기본 정보 그리고 데이터 블록이 복수의 데이터 노드(400) 중 어떤 데이터 노드(400)에 저장되어 있는지에 대한 메타 데이터 등을 저장하고 관리한다. 아울러, 네임 노드(300)는 맵 리듀스 응용과 관련된 복수의 데이터 블록 중에서 로컬 디스크(410)와 인메모리 캐시(420)에 저장된 데이터 블록, 특히 HDFS에 저장된 정보에 관한 정보를 관리한다. 여기서, 복수의 데이터 블록 중에서 인메모리 캐시(420)에 저장된 데이터 블록에 관한 정보는, 복수의 데이터 블록의 개수 대비 인메모리 캐시(420)가 저장하는 데이터 블록의 개수 비율을 포함할 수 있으나 이에 한정되는 것은 아니다. 또한, 네임 노드(300)는 각 맵 리듀스 작업에 대해 인메모리 캐시(420)에 저장된 입력 데이터의 비율을 스케줄링 장치(100)에 주기적으로 전달한다. 한편, 네임 노드(300)에 포함된 캐시 관리부(200) 및 가상 머신 토폴로지(210)에 대해서는 후술하기로 한다.
- [0024] 컨테이너(500)는 태스크를 실행하는 구성이다. 컨테이너(500)가 태스크를 실행하는 경우, 컨테이너(500)는 해당 태스크가 처리해야 하는 데이터 블록을 인메모리 캐시(420)에 저장(caching)하라는 요청을 네임 노드(300)에 전달한다. 이때, 이러한 요청은 해당 데이터 블록이 이미 인메모리 캐시(420)에 저장되어 있는 경우에도 네임 노드(300)에게 전달된다.

- [0025] 스케줄링 장치(100)는 네임 노드(300)로부터 맵 리듀스 작업과 관련된 전체 데이터 블록의 개수 중에서 인메모리 캐시(420)에 저장된 데이터 블록의 개수의 비율을 수신한다.
- [0026] 다음으로, 스케줄링 장치(100)는 캐시 로컬 태스크로 스케줄링 되기까지 대기하는 최대 대기 시간인 제1 대기 시간, 데이터 로컬 태스크로 스케줄링 되기까지 대기하는 최대 대기 시간인 제2 대기 시간이 경과하였는지 여부를 기초로 태스크를 스케줄링한다. 여기서, 제1 대기 시간은 제2 대기 시간보다 짧을 수 있다.
- [0027] 이러한 스케줄링 장치(100)는 본 발명의 실시예에 따라 여러 맵 리듀스 작업을 동시에 실행할 때 가상화 환경과 캐시 지역성을 고려하여 태스크를 스케줄링할 수 있다. 구체적으로, 스케줄링 장치(100)는 가상 머신 토폴로지를 고려하여 동일한 물리 머신(Physical Machine) 내의 가상 머신에 캐시되어 있거나 저장되어 있는 데이터 블록을 활용할 수 있도록 태스크를 스케줄링할 수 있다.
- [0028] 스케줄링 장치(100)가 태스크를 스케줄링하는 보다 자세한 과정은 도 2 및 도 3에서 보다 자세하게 설명하기로 하되, 이러한 스케줄링 장치(100)는 이러한 과정을 수행하도록 프로그램된 명령어를 저장하는 메모리, 그리고 이러한 명령어를 수행하는 마이크로프로세서에 의하여 구현 가능하다.
- [0029] 캐시 관리부(200)는 인메모리 캐시(420)에 데이터 블록을 추가적으로 저장할 공간이 없을 때 맵 리듀스 애플리케이션의 인메모리 캐시(420)에 대한 친밀도를 기초로 인메모리 캐시(420)에 기 저장된 데이터 블록을, 추가하고자 하는 데이터 블록으로 교체하여 저장한다.
- [0030] 또한, 캐시 관리부(200)는 인메모리 캐시(420)에 데이터 블록을 추가할 때 데이터 블록이 인메모리 캐시(420)에 저장될 확률을 고려하여 추가한다. 이 저장될 확률은 맵 리듀스 애플리케이션의 캐시 친밀도(Affinity)와 인메모리 캐시(420)의 크기, 애플리케이션의 입력 데이터 크기에 의해 결정된다.
- [0031] 가상 머신 토폴로지(210)는 네임 노드(300) 내에 포함될 수 있으며, 가상화 환경의 복수의 가상 머신으로 이루어지고, 이러한 가상 머신에 데이터 블록이 캐시되거나 저장될 수 있다.
- [0032] 도 2는 도 1의 HDFS(10)에서 수행되는 가상화 환경을 고려한 스케줄링 장치(100)의 태스크 스케줄링을 예시적으로 설명하는 도면이다.
- [0033] 도 2에 도시한 바와 같이, 가상화 환경은 복수의 가상 머신, 예를 들어 VM1, VM2, VM3, VM4, VM5 및 VM6을 포함하고, 복수의 가상 머신들 중 일부는 물리 머신(PM1, PM2, PM3)에 포함될 수 있다. 그리고, 일부 물리 머신들, 예를 들어 PM1 및 PM2는 랙(Rack), 예를 들어 Rack1에 포함될 수 있다.
- [0034] 맵 리듀스 작업은 태스크가 실행되는 가상 머신, 예를 들어 VM1의 메모리에 입력 데이터가 캐시되어 있을 때 실행 시간이 가장 짧다. 이러한 상태를 VM-ML (Virtual Machine Memory Locality)이라 칭하기로 한다.
- [0035] 하지만 VM-ML로 태스크를 스케줄링 할 수 없을 때 입력 데이터가 동일한 PM에서 실행중인 다른 VM의 메모리에 캐시되어 있다면 디스크의 I/O 없이 VM간 통신만으로 캐시된 입력 데이터를 활용할 수 있으므로 적은 오버헤드로 태스크가 실행중인 VM의 메모리에 캐시된 입력 데이터를 활용하는 것과 비슷한 성능 효과를 갖는다. 이러한 상태를 PM-ML(Physical Machine Memory Locality)이라 칭하기로 한다.
- [0036] 또한, 같은 랙(Rack)에서 실행중인 VM의 메모리에 캐시된 입력 데이터도 PM간 통신만으로 디스크의 I/O 없이 캐시된 입력 데이터를 활용할 수 있기 때문에, 디스크에서 직접 데이터를 읽는 경우보다 실행 시간이 짧아지게 된다. 이러한 상태를 RM-ML(Rack Machine Memory Locality)이라 칭하기로 한다.
- [0037] 따라서, 태스크 스케줄링 시 태스크 실행 시간이 가장 짧은 VM-ML로 태스크를 스케줄링 하도록 하며, VM-ML로 스케줄링 하기 위해 대기하는 시간이 설정된 시간을 초과하게 되면, PM-ML로 스케줄링을 시도하게 된다. PM-ML로 스케줄링 하기 위해 대기하는 시간이 설정된 시간을 초과하게 되면 RM-ML로 태스크 스케줄링을 시도한다.
- [0038] 만약, 메모리에 캐시된 입력 데이터가 없거나, RM-ML로 태스크를 스케줄링 하기 위해 대기하는 시간이 설정된 시간을 초과하게 되면 태스크가 실행되는 VM의 디스크에 저장된 입력 데이터를 활용하도록 태스크 스케줄링을 시도한다. 이러한 상태를 VM-DL(Virtual Machine Disk Locality)이라 칭하기로 한다.
- [0039] 또한, PM-ML과 마찬가지로 같은 PM에서 서비스중인 다른 VM의 디스크에서 읽은 데이터를 활용하는 경우, 동일한 랙(Rack)의 다른 PM에서 서비스중인 VM의 디스크에 저장된 데이터를 활용하는 경우보다 오버헤드가 적다. 따라서 VM-DL로 스케줄링 하기 위해 대기하는 시간이 설정된 시간을 초과하는 경우, 동일한 PM의 다른 VM의 디스크에 저장된 입력 데이터를 활용하도록 태스크 스케줄링을 시도한다. 이러한 상태를 PM-DL(Physical Machine

Disk Locality)이라 칭하기로 한다.

- [0040] 마지막으로, 스케줄링 하지 못한 경우, 동일한 랙의 다른 PM에서 서비스중인 VM의 디스크에 저장된 입력 데이터를 활용하도록 태스크를 스케줄링 한다. 이러한 상태를 RM-DL(Rack Machine Disk Locality)이라 칭하기로 한다.
- [0041] 요약하면, 본 발명의 실시예에 따른 가상화 환경에서의 스케줄링은 다음과 같은 순서로 진행될 수 있다.
- [0042] VM-ML -> PM-ML -> RM-ML -> VM-DL -> PM-DL -> RM-DL
- [0043] 도 3은 본 발명의 실시예에 따른 분산 파일 시스템, 예컨대 인메모리 캐시 기반의 HDFS를 포함하는 빅데이터 처리 플랫폼에서의 태스크 스케줄링 과정을 예시적으로 설명하는 흐름도이다.
- [0044] 도 3을 참조하면, 먼저 스케줄링 장치(100)는 맵 리듀스 작업이 실행될 때 네임 노드(300)로부터 노드의 인메모리 캐시(420)가 저장하는 데이터 블록에 관한 정보를 수신한다(S100, S110). 여기서, 인메모리 캐시(420)가 저장하는 데이터 블록에 관한 정보란, 예컨대 맵 리듀스 작업과 관련된 전체 데이터 블록의 개수 중에서 인메모리 캐시(420)가 저장하는 데이터 블록의 개수의 비율일 수 있다.
- [0045] 다음으로, 스케줄링 장치(100)는 이러한 데이터 블록에 관한 정보를 기초로 맵 리듀스 작업에 이용되는 데이터 블록이 인메모리 캐시(420)에 존재하는지 여부에 따라 맵 리듀스 작업의 태스크가 실행되는 가상 머신에 캐시되어 있는 데이터 블록을 활용할지, 또는 가상 머신의 디스크에 저장되어 있는 데이터 블록을 활용할지를 결정한 후 맵 리듀스 작업의 태스크를 스케줄링할 수 있다.
- [0046] 이때, 스케줄링 장치(100)는 캐시 로컬 태스크로 스케줄링되기까지 대기하는 시간의 한계인 제1 대기 시간을 산정한다. 제1 대기 시간은 기 지정된 최대 대기 시간 C 에 네임 노드(300)로부터 수신한 비율을 곱한 값으로 산정될 수 있다.
- [0047] 단계(S120)에서의 판단 결과, 맵 리듀스 작업에 이용되는 데이터 블록이 인메모리 캐시에 존재하면, 스케줄링 장치(100)는 맵 리듀스 작업의 태스크가 실행되는 가상 머신에 캐시되어 있는 데이터 블록을 활용하여 맵 리듀스 작업의 태스크를 스케줄링할 수 있다(S130).
- [0048] 여기서, 맵 리듀스 작업에 이용되는 데이터 블록이 인메모리 캐시에 존재하는 경우에는, 상술한 바와 같이 VM-ML -> PM-ML -> RM-ML의 순서로 태스크를 스케줄링할 수 있다.
- [0049] 이때, 스케줄링하는 단계(S130)는, 맵 리듀스 작업의 태스크가 실행되는 복수의 물리 머신 내의 가상 머신에 캐시되어 있는 데이터 블록을 활용하여 맵 리듀스 작업의 태스크를 스케줄링할 수 있다. 예컨대, 도 2에 도시한 바와 같이, 복수의 물리 머신들 중 각각의 물리 머신, 예를 들어 PM1 내에 복수의 가상 머신, 예를 들어 VM1 및 VM2가 상주(co-resident)할 경우, 맵 리듀스 작업의 태스크가 실행되는 가상 머신(VM1)이 상주하는 물리 머신(PM1)과 다른 물리 머신(예를 들어, PM2) 내의 가상 머신(예를 들어, VM3)에 캐시되어 있는 데이터 블록을 활용하여 맵 리듀스 작업의 태스크를 스케줄링할 수 있다.
- [0050] 한편, 맵 리듀스 작업의 데이터 블록이 인메모리 캐시(420)에 존재하지 않거나, 스케줄링하는 단계(S130)의 대기 시간이 기 설정된 시간을 초과하면(S140), 스케줄링 장치(100)는 가상 머신의 디스크에 저장되어 있는 데이터 블록을 활용하여 맵 리듀스 작업의 태스크를 스케줄링할 수 있다(S150).
- [0051] 여기서, 맵 리듀스 작업에 이용되는 데이터 블록이 인메모리 캐시에 존재하지 않을 경우에는, 상술한 바와 같이 VM-DL -> PM-DL -> RM-DL의 순서로 태스크를 스케줄링할 수 있다.
- [0052] 이상 설명한 바와 같이 본 발명의 실시예에 의하면, 분산 파일 시스템을 기반으로 하는 빅데이터 처리 플랫폼, 특히 인메모리 캐시 기반의 HDFS를 포함하는 빅데이터 처리 플랫폼에서 가상화 환경과 캐시 지역성을 함께 고려하여 맵 리듀스 작업의 태스크를 스케줄링함으로써, 맵 리듀스의 작업이 HDFS에 캐시된 입력 데이터를 효율적으로 사용하도록 구현한 것이다.
- [0053] 한편, 첨부된 블록도의 각 블록과 흐름도의 각 단계의 조합들은 컴퓨터 프로그램 인스트럭션들에 의해 수행될 수도 있다. 이들 컴퓨터 프로그램 인스트럭션들은 범용 컴퓨터, 특수용 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비의 프로세서에 탑재될 수 있으므로, 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비의 프로세서를 통해 수행되는 그 인스트럭션들이 블록도의 각 블록에서 설명된 기능들을 수행하는 수단을 생성하게 된다.

[0054] 이들 컴퓨터 프로그램 인스트럭션들은 특정 방식으로 기능을 구현하기 위해 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비를 지향할 수 있는 컴퓨터 이용 가능 또는 컴퓨터 판독 가능 기록매체(또는 메모리) 등에 저장되는 것도 가능하므로, 그 컴퓨터 이용 가능 또는 컴퓨터 판독 가능 기록매체(또는 메모리)에 저장된 인스트럭션들은 블록도의 각 블록에서 설명된 기능을 수행하는 인스트럭션 수단을 내포하는 제조 품목을 생산하는 것도 가능하다.

[0055] 그리고, 컴퓨터 프로그램 인스트럭션들은 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비 상에 탑재되는 것도 가능하므로, 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비 상에서 일련의 동작 단계들이 수행되어 컴퓨터로 실행되는 프로세스를 생성해서 컴퓨터 또는 기타 프로그램 가능한 데이터 프로세싱 장비를 수행하는 인스트럭션들은 블록도의 각 블록에서 설명된 기능들을 실행하기 위한 단계들을 제공하는 것도 가능하다.

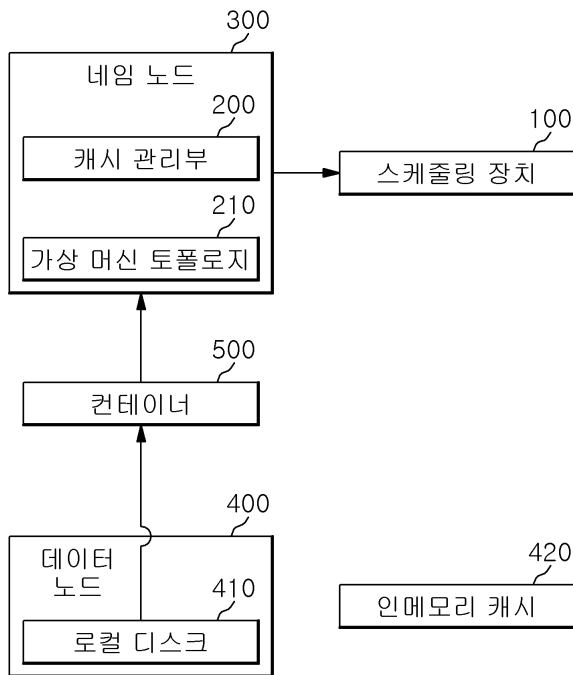
[0056] 또한, 각 블록은 특정된 논리적 기능(들)을 실행하기 위한 적어도 하나 이상의 실행 가능한 인스트럭션들을 포함하는 모듈, 세그먼트 또는 코드의 일부를 나타낼 수 있다. 또, 몇 가지 대체 실시 예들에서는 블록들에서 언급된 기능들이 순서를 벗어나서 발생하는 것도 가능함을 주목해야 한다. 예컨대, 잇달아 도시되어 있는 두 개의 블록들은 사실 실질적으로 동시에 수행되는 것도 가능하고 또는 그 블록들이 때때로 해당하는 기능에 따라 역순으로 수행되는 것도 가능하다.

부호의 설명

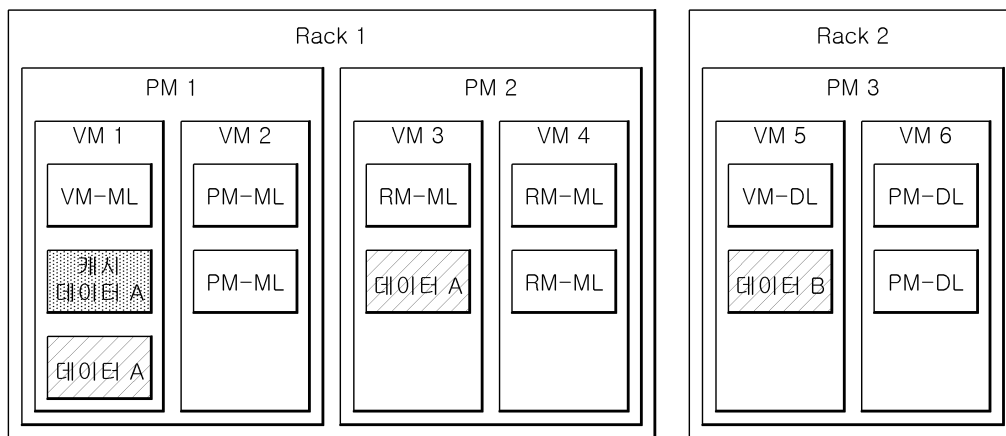
[0057] 100: 스케줄링 장치
200: 캐시 관리부
210: 가상 머신 토폴로지
300: 네임 노드
400: 데이터 노드
410: 로컬 디스크
420: 인메모리 캐시
500: 컨테이너

도면

도면1



도면2



도면3

