(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년07월17일
(11) 등록번호 10-1285078
(24) 등록일자 2013년07월05일

(51) 국제특허분류(Int. Cl.)
G06F 9/28 (2006.01) *G06F 9/38* (2006.01)
G06F 15/16 (2006.01) *G06F 11/30* (2006.01)
(21) 출원번호 10-2009-0126035
(22) 출원일자 2009년12월17일
심사청구일자 2009년12월17일
(65) 공개번호 10-2011-0069338
(43) 공개일자 2011년06월23일
(56) 선행기술조사문헌
In CIDR Conference, FRANKLIN, M.J., et al. Continuous analytics: Rethinking query processing in a network-effect world, January 2009 (2009.01.)*
IN: 2009 IEEE Cluster Computing and Workshops, PALLICKARA, S. et al. Granules: A lightweight, streaming runtime for cloud computing with support, for Map-Reduce (2009. 09.04)
KR1020090066010 A
KR100919370 B1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
한국전자통신연구원
대전광역시 유성구 가정로 218 (가정동)
(72) 발명자
이명철
대전광역시 유성구 배울2로 24, 중앙하이츠빌 308동 1401호 (관평동)
이미영
대전광역시 유성구 어은로 57, 103동 705호 (어은동, 한빛아파트)
(74) 대리인
한양특허법인

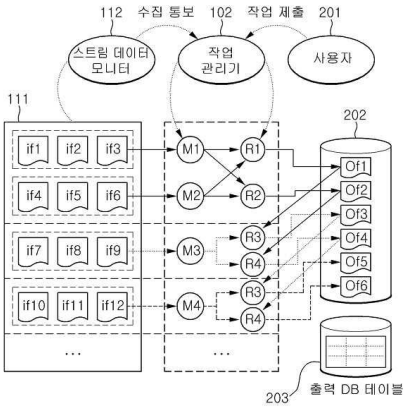
전체 청구항 수 : 총 15 항 심사관 : 황승희

(54) 발명의 명칭 **스트림 데이터에 대한 점진적인 맵리듀스 기반 분산 병렬 처리 시스템 및 방법**

(57) 요 약

대용량의 데이터를 다수의 컴퓨팅 노드를 이용하여 MapReduce 방식으로 분산 병렬 처리하는 시스템으로서, 이미 수집되어 있는 대용량 저장 데이터는 물론 분산 병렬 처리 작업이 수행되는 동안에도 연속적으로 수집되는 대량의 스트림 데이터에 대해서 점진적인 MapReduce 기반 분산 병렬 처리 기능을 제공하기 위한 분산 병렬 처리 시스템이 제공된다.

대 표 도 - 도2



이 발명을 지원한 국가연구개발사업

과제고유번호	2007-S-016-03
부처명	지식경제부
연구사업명	IT성장동력기술개발
연구과제명	저비용 대규모 글로벌 인터넷 서비스 솔루션 개발
주관기관	한국전자통신연구원
연구기간	2007-03-01 ~ 2012-02-29

특허청구의 범위

청구항 1

입력 데이터 저장 위치에 수집되는 추가 데이터의 여부를 주기적으로 모니터링하는 스트림 데이터 모니터;

상기 스트림 데이터 모니터의 모니터링 결과에 따라 하나 이상의 추가 태스크를 생성하고, 기존 태스크로부터 출력된 최종 결과와 상기 하나 이상의 추가 태스크로부터 생성된 중간 결과를 병합하여 새로운 최종 결과를 출력하는 작업 관리기; 및

상기 하나 이상의 태스크 또는 상기 기존 태스크의 동작을 관리하는 하나 이상의 태스크 관리기를 포함하고,

상기 작업 관리기는

수집된 추가 데이터를 처리하여 상기 중간 결과를 출력하기 위한 하나의 맵 태스크; 및 상기 하나의 맵 태스크로부터 출력되는 상기 중간 결과를 처리하기 위한 하나 이상의 리듀스 태스크를 생성하여 새로 수집된 추가 데이터를 분산 병렬 처리하도록 제어하는 것을 특징으로 하는 분산 병렬 처리 시스템.

청구항 2

청구항 1에 있어서, 상기 작업 관리기는,

상기 하나 이상의 리듀스 태스크는 기존 리듀스 태스크와 동일 개수로 생성되는 분산 병렬 처리 시스템.

청구항 3

청구항 1에 있어서,

상기 하나 이상의 리듀스 태스크는 상기 하나의 맵 태스크로부터 출력되는 상기 중간 결과와 상기 기존 리듀스 태스크로부터 출력된 최종 결과를 병합하여 상기 새로운 최종 결과를 출력하는 분산 병렬 처리 시스템.

청구항 4

청구항 1에 있어서,

상기 하나의 맵 태스크는 기존 맵 태스크와 독립적으로 동작하는 분산 병렬 처리 시스템.

청구항 5

청구항 1에 있어서,

상기 하나 이상의 리듀스 태스크는 상기 하나 이상의 맵 태스크 또는 상기 기존 리듀스 태스크의 동작이 완료된 이후에 동작하는 분산 병렬 처리 시스템.

청구항 6

청구항 1에 있어서,

상기 스트림 데이터 모니터는 상기 입력 데이터 저장 위치에 수집된 상기 추가 데이터의 처리 시간으로부터 로그 파일을 생성하고, 상기 로그 파일을 참조하여 상기 처리 시간 이후에 수집된 데이터를 추가 데이터로 인식하는 분산 병렬 처리 시스템.

청구항 7

청구항 1에 있어서,

상기 하나 이상의 태스크 또는 상기 기존 태스크로부터 각각 생성된 최종 결과를 주기적으로 병합하는 최종 결과 병합기를 더 포함하는 분산 병렬 처리 시스템.

청구항 8

삭제

청구항 9

분산 병렬 처리 시스템이 수집되는 추가 데이터를 모니터링한 결과에 따라 하나 이상의 추가 태스크를 생성하는 단계; 및

상기 분산 병렬 처리 시스템이 기존 태스크로부터 출력된 최종 결과와 상기 하나 이상의 추가 태스크로부터 생성된 중간 결과를 병합하여 새로운 최종 결과를 출력하는 단계를 포함하고,

상기 하나 이상의 추가 태스크를 생성하는 단계는

상기 추가 데이터를 처리하여 상기 중간 결과를 출력하기 위한 하나의 맵 태스크와 상기 하나의 맵 태스크로부터 출력되는 상기 중간 결과를 처리하기 위한 하나 이상의 리듀스 태스크를 생성하여, 새로 수집된 추가 데이터를 분산 병렬 처리하도록 제어하는 것을 특징으로 하는 분산 병렬 처리 방법.

청구항 10

청구항 9에 있어서, 상기 하나 이상의 추가 태스크를 생성하는 단계는,

상기 하나 이상의 리듀스 태스크를 기존 리듀스 태스크와 동일 개수로 생성하는 단계를 포함하는 분산 병렬 처리 방법.

청구항 11

청구항 9에 있어서, 상기 새로운 최종 결과를 출력하는 단계는,

상기 하나의 맵 태스크가 상기 추가 데이터를 처리하여 상기 중간 결과를 출력하는 단계; 및

상기 하나 이상의 리듀스 태스크가 상기 하나의 맵 태스크로부터 출력된 상기 중간 결과와 상기 기존 리듀스 태스크로부터 출력된 최종 결과를 병합하여 상기 새로운 최종 결과를 출력하는 단계를 포함하는 분산 병렬 처리 방법.

청구항 12

청구항 9에 있어서,

상기 기존 태스크로부터 출력된 최종 결과와 상기 하나 이상의 추가 태스크로부터 출력된 새로운 최종 결과를 주기적으로 병합하여 하나의 최종 결과를 출력하는 단계를 포함하는 분산 병렬 처리 방법.

청구항 13

청구항 12에 있어서, 상기 하나의 최종 결과를 출력하는 단계는,

상기 기존 태스크 또는 상기 하나 이상의 추가 태스크로부터 출력된 하나 이상의 최종 결과의 개수를 설정값과 비교하는 단계; 및

비교 결과에 따라 상기 하나 이상의 최종 결과를 병합하거나 또는 일정 시간 동안 수면하는 단계를 포함하는 분산 병렬 처리 방법.

청구항 14

청구항 13에 있어서, 상기 하나의 최종 결과를 출력하는 단계는,

상기 하나 이상의 최종 결과의 개수가 상기 설정값보다 적으면, 일정 시간 동안 수면하는 분산 병렬 처리 방법.

청구항 15

청구항 13에 있어서, 상기 하나의 최종 결과를 출력하는 단계는,

상기 하나 이상의 최종 결과의 개수가 상기 설정값보다 많거나 동일하면, 상기 하나 이상의 최종 결과를 병합하는 분산 병렬 처리 방법.

청구항 16

청구항 9에 있어서,

수집되는 추가 데이터의 처리 시간으로부터 로그 파일을 생성하고, 상기 로그 파일을 참조하여 상기 처리 시간 이후에 수집되는 데이터를 추가 데이터로 인식하는 단계를 더 포함하는 분산 병렬 처리 방법.

명 세 서

발명의 상세한 설명

기 술 분 야

[0001] 본 발명은 스트림 데이터 처리 시스템 및 처리 방법에 관한 것으로서, 더욱 상세하게는 대용량의 데이터를 다수의 컴퓨팅 노드를 이용하여 MapReduce 방식으로 분산 병렬 처리하는 시스템 및 이의 분산 병렬 처리 방법에 관한 것이다.

[0002] 본 발명은 지식경제부의 IT성장동력기술개발사업의 일환으로 수행한 연구로부터 도출된 것이다[과제관리번호: 2007-S-016-03, 과제명: 저비용 대규모 글로벌인터넷 서비스 솔루션 개발].

배 경 기 술

[0003] 웹 2.0의 등장으로 인터넷 서비스가 공급자 중심에서 사용자 중심으로 패러다임이 이동함에 따라 UCC, 개인화 서비스와 같은 인터넷 서비스 시장이 급속도로 증가하고 있다. 이러한 패러다임의 변화로 사용자에 의해서 생성되고 인터넷 서비스를 위해 수집, 처리, 그리고 관리해야 하는 데이터의 양이 빠르게 증가하고 있다.

[0004] 이와 같은 대용량 데이터의 수집, 처리 및 관리를 위하여, 현재 많은 인터넷 포탈에서 저비용으로 대규모 클러스터를 구축하여 대용량 데이터 분산 관리 및 작업 분산 병렬 처리하는 기술에 대하여 많은 연구를 하고 있으며, 작업 분산 병렬 처리 기술 중에서 미국 Google 사의 MapReduce 모델이 대표적인 작업 분산 병렬 처리 방법 중에 하나로 주목을 받고 있다.

[0005] MapReduce 모델은 Google 사에서 저비용 대규모 노드로 구성된 클러스터 상에 저장된 대용량 데이터에 대한 분산 병렬 연산을 지원하기 위하여 제안한 분산 병렬 처리 프로그래밍 모델이다.

[0006] MapReduce 모델 기반의 분산 병렬 처리 시스템으로는, Google의 MapReduce 시스템, Apache Software Foundation의 Hadoop MapReduce 시스템과 같은 분산 병렬 처리 시스템이 있다.

[0007] 이들 MapReduce 모델 기반 분산 병렬 처리 시스템들은 기본적으로 이미 수집되어 저장된 대용량 데이터에 대한 주기적인 오프라인 배치 처리 만을 지원하고 있으며, 연속적으로 수집되는 스트림 데이터에 대한 실시간 처리에 대해서는 별로 고민하고 있지 않아서, 새로 수집되는 입력 데이터에 대해서는 주기적으로 배치 처리를 해야 하는 실정이다.

[0008] 또한, MapReduce 모델 기반 분산 병렬 처리 시스템을 사용하는 대부분의 인터넷 포탈의 경우, 이와 같이 대용량으로 수집되는 인터넷 데이터, UCC, 개인화 서비스 데이터 등에 대해서 인덱스 구축을 통해서 사용자에게 빠른 검색 기능을 제공하거나, 의미 있는 통계 정보를 추출해서 마케팅에 이용하는 등의 데이터 처리 작업을 주로 요구하고 있다.

[0009] 이와 같이 인터넷 포탈에서 제공하는 서비스들은 대체로 시간이 많이 걸리더라도 정확한 결과를 찾아 주는 정확 검색 보다는 일정 허용 범위 내에서 정확한 결과에 근접하는 결과를 빨리 찾아 주는 유사도 검색을 주로 지원하고 있어서, 실시간 데이터 처리가 더욱 요구되는 환경이라고 할 수 있다.

[0010] 따라서, 인터넷 서비스를 제공하는 인터넷 포탈 입장에서는 엄청난 속도로 수집되는 방대한 양의 스트림 데이터로부터 가능한 빨리 의미 있는 정보를 추출하여 사용자에게 서비스하는 능력이 기업의 경쟁력이 된다. 그러나, 기존 시스템들이 제공하는 배치 처리 방식의 분산 병렬 처리 모델로는 인터넷 포탈이 원하는 방대한 양의 스트림 데이터에 대한 실시간 처리가 거의 불가능한 것이 현실이다.

발명의 내용

해결 하고자하는 과제

[0011] 본 발명은 연속적으로 수집이 되는 대용량 스트림 데이터에 대하여 점진적인 MapReduce 기반의 분산 병렬 처리 기술을 제공하여 거의 실시간에 가까운 고속 데이터 처리 시스템 및 기능을 제공하는데 그 목적이 있다.

과제 해결수단

[0012] 이러한 목적을 달성하기 위한 본 발명에 따른 분산 병렬 처리 시스템은, 입력 데이터 저장 위치에 수집되는 추가 데이터의 여부를 주기적으로 모니터링하는 스트림 데이터 모니터 및 상기 스트림 데이터 모니터의 모니터링 결과에 따라 하나 이상의 추가 태스크를 생성하고, 기존 태스크로부터 출력된 최종 결과와 상기 하나 이상의 추가 태스크로부터 생성된 중간 결과를 병합하여 새로운 최종 결과를 출력하는 작업 관리기를 포함한다.

[0013] 본 발명에 따른 분산 병렬 처리 방법은, 수집되는 추가 데이터를 모니터링한 결과에 따라 하나 이상의 추가 태스크를 생성하는 단계 및 기존 태스크로부터 출력된 최종 결과와 상기 하나 이상의 추가 태스크로부터 생성된 중간 결과를 병합하여 새로운 최종 결과를 출력하는 단계를 포함한다.

효 과

[0014] 본 발명에 따른 분산 병렬 처리 시스템 및 처리 방법은 다음과 같은 효과를 기대할 수 있다.

[0015] 첫째, 실시간에 가까운 고속 데이터 처리를 수행할 수 있다.

[0016] 둘째, 연속적으로 수집되는 스트림에 대한 처리를 수행할 수 있다.

[0017] 셋째, 대용량 스트림 데이터에 대한 처리를 수행할 수 있다.

발명의 실시를 위한 구체적인 내용

[0018] 본 발명과 본 발명의 동작상의 이점 및 본 발명의 실시 예에 의하여 달성되는 목적을 충분히 이해하기 위해서는 본 발명의 실시 예를 예시하는 첨부도면 및 첨부도면에 기재된 내용을 참조하여야만 한다.

[0019] 본 발명은 MapReduce 방식의 분산 병렬 처리 모델을 지원하는 다중 노드로 구성된 클러스터 상에서의 대용량 데이터에 대한 작업 분산 병렬 처리 시스템에서, 기존에 수집되어 저장된 대용량 데이터에 대한 분산 병렬 처리뿐만 아니라, 연속적으로 수집되는 대용량 스트림 데이터에 대해서도 점진적으로 분산 병렬 처리 기능을 제공하여, 연속적으로 수집되는 대용량 스트림 데이터에 대해서 거의 실시간 분산 병렬 처리 기능을 제공하는 방법에 관한 것이다.

[0020] 이하, 첨부한 도면을 참조하여 본 발명의 바람직한 실시 예를 설명함으로써, 본 발명을 상세히 설명한다. 각 도면에 제시된 동일한 참조부호는 동일한 부재를 나타낸다.

[0021] 도 1은 본 발명에 따른 분산 병렬 처리 시스템의 시스템 구성도이다.

[0022] 도 1에서 도시된 바와 같이, 본 발명의 분산 병렬 처리 시스템은 작업 관리기(102), 스트림 데이터 모니터(112), 최종 결과 병합기(113) 및 하나 이상의 태스크 관리기(103, 107)를 포함할 수 있다.

[0023] 작업 관리기(102)는 작업 관리를 담당하는 노드에서 수행되어 전체 작업 처리 과정을 제어하고 관리할 수 있다.

[0024] 스트림 데이터 모니터(112)는 새로운 데이터가 수집되었는지 주기적으로 검사하는 역할을 수행할 수 있다.

[0025] 스트림 데이터 모니터(112)는 입력 데이터 저장 위치(111)에 새로운 데이터, 즉 추가 스트림 데이터가 수집되었는지 주기적으로 검사하고, 검사 결과에 따른 정보를 작업 관리기(102)에 통보할 수 있다.

[0026] 여기서, 스트림 데이터 모니터(112)는 입력 데이터 저장 위치(111)에 입력되는 새로운 데이터를 관리하기 위하여 입력된 새로운 데이터를 처리한 마지막 시간, 즉 후술될 작업 관리기(102)에 의해 입력 데이터를 분산 병렬 처리하여 완료한 시간을 로깅하여 로그 파일을 생성하고, 생성된 로그 파일을 참조하여 그 시간(즉, 처리 시간) 이후에 입력 데이터 저장 위치(111)로 수집된 데이터만을 새로운 데이터로 인식할 수 있다.

[0027] 작업 관리기(102)는 스트림 데이터 모니터(112)로부터의 통보에 따라 하나 이상의 추가 태스크, 예컨대 새로운 맵(이하, Map) 태스크와 리듀스(이하, Reduce) 태스크를 생성하여 새로 수집된 추가 데이터를 분산 병렬 처리하

도록 제어할 수 있다.

[0028] 최종 결과 병합기(113)는 Reduce 태스크가 생성한 여러 버전의 최종 결과를 주기적으로 병합할 수 있다.

[0029] 최종 결과 병합기(113)는 출력 데이터 저장 위치에 여러 버전의 출력 결과가 저장되어 있는 경우, 이를 주기적으로 통합하여 하나의 버전의 출력 결과로 만들어 주는 역할을 수행하고, 수행 결과를 작업 관리기(102)에 통보할 수 있다.

[0030] 작업 관리기(102)는 새로운 Reduce 태스크 생성 시에 이전 수행 결과를 제공할 때 최종 결과 병합기(113)로부터 출력된 병합에 의해 생성된 최종 결과가 있으면 해당 파일의 위치를 제공할 수 있다.

[0031] 하나 이상의 태스크 관리기(103, 107)는 각 태스크 관리기에 할당되는 다수개의 Map 태스크를 실제로 수행하는 다수개의 Map 태스크 수행기(104, 108) 및 다수개의 Reduce 태스크를 실제로 수행하는 다수개의 Reduce 태스크 수행기(105, 109)를 각각 포함할 수 있다.

[0032] 다수개의 Map 태스크 수행기(104, 108) 또는 다수개의 Reduce 태스크 수행기(105, 109)는 Map 태스크 또는 Reduce 태스크가 할당이 되어 수행이 되는 과정에서 생성이 되며, 수행이 완료되면 메모리에서 제거될 수 있다.

[0033] 본 발명이 제안하는 스트림 데이터에 대한 점진적인 MapReduce 방식의 분산 병렬 처리 서비스를 제공하는 방법은 도 2에서 보는 바와 같다.

[0034] 도 2는 본 발명에 따른 분산 병렬 처리 방법의 동작을 나타낸 예시도이다.

[0035] 도 2를 참조하면, 사용자(201)가 '입력 데이터 저장 위치', '출력 데이터 저장 위치', '사용자 정의 Map 함수', '사용자 정의 Reduce 함수', '사용자 정의 Update 함수', 'Reduce 태스크 개수', '처리 완료 입력 삭제 여부', '작업 수행 종료 시간' 등으로 구성이 된 MapReduce 기반 분산 병렬 처리 작업을 작업 관리기(102)에 제출하여 분산 병렬 처리를 요청한다.

[0036] 작업 관리기(102)는 입력 데이터 저장 위치(111) 내의 파일 목록을 읽어 와서 전체 입력 데이터 크기를 산출한 후 적절한 개수의 Map 태스크(M1, M2)를 생성하여 태스크 수행 노드의 Map 태스크 수행기에 할당하여 처리가 되도록 한다.

[0037] 또한, 작업 관리기(102)는 사용자가 입력한 Reduce 태스크 개수만큼의 Reduce 태스크(R1, R2)를 생성하여 태스크 수행 노드의 Reduce 태스크 수행기에 할당하여 처리가 되도록 한다.

[0038] Map 태스크(M1, M2)는 할당 받은 입력 파일을 처리하여 중간 결과 파일을 생성한다.

[0039] 이때, 각 Map 태스크가 생성한 중간 결과는 사용자가 등록한 파티션 함수에 따라서 여러 Reduce 태스크에 고르게 분배가 된다.

[0040] 각 Map 태스크로부터 중간 결과를 복사한 Reduce 태스크(R1, R2)는 처리를 완료한 후 최종 결과를 사용자가 명시한 출력 데이터 저장 위치(215) 내의 파일(of1, of2)로 작성하거나, 출력 DB 테이블(203)에 삽입한다.

[0041] 스트림 데이터 모니터(112)는 입력 데이터 저장 위치(111)에 현재 처리 중인 입력 파일 외에 추가적인 파일들이 수집이 되었는지 주기적으로 모니터링한다.

[0042] 모니터링 결과에 따라 적절한 용량의 새로운 입력 데이터가 수집이 되면 작업 관리기(102)에 통보하고, 작업 관리기(102)는 해당 추가 입력 파일들을 처리하기 위한 새로운 Map 태스크(M3)를 생성하여 태스크 수행 노드의 Map 태스크 수행기에 할당하여 처리하게 한다.

[0043] 또한, 작업 관리기(102)는 해당 Map 태스크(M3)의 중간 결과를 처리하기 위한 Reduce 태스크(R3, R4)를 생성하여 태스크 수행 노드의 Reduce 태스크 수행기에 할당하여 처리하게 한다.

[0044] 이때 새로 생성되는 Reduce 태스크(R3, R4)는 기존 Reduce 태스크(R1, R2)와 같은 개수로 생성이 된다.

[0045] 기존 Reduce 태스크(R1, R2)는 기존 Map 태스크(M1, M2)가 생성한 중간 결과 파일로부터 첫번째 최종 결과를 생성하여 출력 데이터 저장 위치(202)의 파일(of1, of2)로 작성하거나 또는 출력 DB 테이블(203)에 삽입하여 저장한다.

[0046] 이후, 새로운 Map 태스크(M3)가 생성이 되면, 새로운 Reduce 태스크(R3, R4)는 해당 Map 태스크(M3)가 생성한

중간 결과와 기존 Map 태스크(M1, M2)가 생성한 중간 결과로부터 기존 Reduce 태스크(R1, R2)에 의해서 만들어진 기존 최종 결과(of1, of2)를 통합하여 새로운 최종 결과(of3, of4)를 생성하고, 생성된 새로운 최종 결과(of3, of4)를 출력 데이터 저장 위치(202)의 파일(of3, of4)로 작성하거나 또는 출력 DB 테이블(203)에 삽입하여 저장한다.

[0047] 또한, 상술한 과정들은 입력 데이터 저장 위치(111)에 새로운 데이터, 즉 추가 파일들이 수집될 때마다 반복하여 수행될 수 있으며, 이에 따라 연속적으로 수집되는 스트림 데이터에 대한 점진적인 MapReduce 기반 분산 병렬 처리 기능을 제공하게 된다.

[0048] 예컨대, 스트림 데이터 모니터(112)가 입력 데이터 저장 위치(111)에 추가적인 파일들이 수집이 되었는지 모니터링 하고, 모니터링 결과에 따라 새로운 입력 데이터가 다시 수집이 되면 작업 관리기(102)에 통보하고, 작업 관리기(102)는 해당 추가 입력 파일들을 처리하기 위한 새로운 Map 태스크(M4)를 생성하여 태스크 수행 노드의 Map 태스크 수행기에 할당하여 처리하게 한다.

[0049] 또한, 작업 관리기(102)는 해당 Map 태스크(M4)의 중간 결과를 처리하기 위한 Reduce 태스크(R5, R6)를 생성하여 태스크 수행 노드의 Reduce 태스크 수행기에 할당하여 처리하게 한다.

[0050] 이때 새로 생성되는 Reduce 태스크(R5, R6)는 기존 Reduce 태스크(R1, R2 또는 R3, R4)와 같은 개수로 생성이 된다.

[0051] 새로운 Reduce 태스크(R5, R6)는 해당 Map 태스크(M4)가 생성한 중간 결과와 기존 Map 태스크(M3)가 생성한 중간 결과로부터 기존 Reduce 태스크(R3, R4)에 의해서 만들어진 기존 최종 결과(of3, of4)를 통합하여 새로운 최종 결과(of5, of6)를 생성하고, 생성된 새로운 최종 결과(of5, of6)를 출력 데이터 저장 위치(202)의 파일(of5, of6)로 작성하거나 또는 출력 DB 테이블(203)에 삽입하여 저장한다.

[0052] 한편, 기존 Map 태스크(M1, M2)와 Reduce 태스크(R1, R2)는 할당 받은 입력 데이터의 처리가 끝나면 바로 종료된다.

[0053] 또한, 새로운 Map 태스크(M3)와 Reduce 태스크(R3, R4) 역시 새로 수집된 입력 데이터(if7, if8, if9)의 처리가 끝나면 바로 종료가 된다.

[0054] 새로운 Map 태스크(M3)는 기존 Map 태스크(M1, M2)의 완료 여부와 상관없이 독립적으로 처리가 시작이 되며, 새로운 Map 태스크(M4)는 기존 Map 태스크(M1, M2, M3)의 완료 여부와 상관없이 독립적으로 처리가 시작된다.

[0055] 그러나, 새로운 Reduce 태스크(R3, R4)는 연관된 Map 태스크(M3)의 중간 결과와 기존 Reduce 태스크(R1, R2)가 생성한 기존 최종 결과를 입력 받아서 수행이 되기 때문에, 항상 Map 태스크(M3)와 기존 Reduce 태스크(R1, R2)의 수행이 완료된 이후에 수행이 시작된다.

[0056] 또한, 새로운 Reduce 태스크(R5, R6)는 연관된 Map 태스크(M4)의 중간 결과와 기존 Reduce 태스크(R3, R4)가 생성한 기존 최종 결과를 입력받아서 수행이 되기 때문에, 항상 Map 태스크(M4)와 기존 Reduce 태스크(R3, R4)의 수행이 완료된 이후에 수행이 시작된다.

[0057] Reduce 태스크(R1, R2, R3, R4, R5, R6)가 출력 데이터 저장 위치에 최종 결과를 생성하는 디렉토리 구성을 살펴 보면 도 3에서 보는 바와 같은 디렉토리 구성으로 생성된다.

[0058] 도 3은 출력 데이터 저장 위치에 최종 결과를 생성하는 디렉토리 구성을 나타낸 예시도이다.

[0059] 도 3을 참조하면, 사용자가 작업 제출 시에 제공한 출력 데이터 저장 위치(202)가 'output_dir'인 경우, 처음 작업 제출 시점에 수행된 Reduce 태스크(R1, R2)의 저장 위치는 첫번째 수행 시간을 나타내는 시점의 타임스탬프 값을 표현하여 예를 들어, output_dir/1254293251990 디렉토리 밑에 각각 output_dir/1254293251990/r1과 output_dir/1254293251990/r2 디렉토리에 저장이 된다.

[0060] 이후 2번째로 수행되는 Reduce 태스크(R3, R4)의 최종 결과는 2번째 수행 시점을 나타내는 output_dir/1254293251991 디렉토리 아래에 저장된다.

[0061] 또한, 3번째로 수행되는 Reduce 태스크(R5, R6)의 최종 결과는 3번째 수행 시점을 나타내는 output_dir/1254293251992 밑에 저장이 되어, 가장 최신의 데이터가 가장 타임스탬프가 큰 값을 갖는 디렉토리에 저장이 되도록 한다.

[0062] 또한, 최종 결과 병합기(도 1의 도면부호 113)는 여러 버전의 최종 결과를 주기적으로 병합하고 해당 병합 시점을 나타내는 디렉토리에 최종 결과를 작성할 수 있다.

[0063] 이때, 이전 버전의 최종 결과는 삭제되고, 새로 생성된 최종 결과는 이후 수행되는 Reduce 태스크에 이전 최종 결과로서 사용이 된다.

[0064] 도 4는 본 발명의 MapReduce 프로그래밍 모델의 예시도이다.

[0065] 본 발명의 MapReduce 프로그래밍 모델은 도 4에서 보는 바와 같이 사용자 정의 Map 함수(401), 사용자정의 Reduce 함수(402) 그리고 사용자정의 업데이트(Update) 함수(403)로 구성된다.

[0066] 본 발명의 MapReduce 프로그래밍 모델은 사용자가 기존 Reduce 함수(402) 처리 결과를 얻어 오는 방법을 명세할 수 있도록 Update 함수(403)를 추가하고, Update 함수(403)의 결과를 Reduce 함수에 넘기기 위하여 old_values 인자(404)를 추가한 프로그래밍 모델이다.

[0067] 본 발명의 MapReduce 프로그래밍 모델을 따르는 분산 병렬 처리 작업은 기본적으로 기존 결과가 있다는 가정 하에 수행될 수 있으며, 기존 결과 파일 또는 기존 결과 DB에서 값을 읽어 오는 방법을 Update 함수 내에 사용자가 제공해 줘야 한다.

[0068] 여기서, 사용자가 Update 함수를 제공하지 않으면, MapReduce 프로그래밍 모델의 Reduce 함수에서는 기존 Reduce 함수 수행 결과 값을 알 수가 없으므로, 언제나 기존 결과 값이 없다고 판단하고 새로운 결과 값을 파일 또는 DB에 덮어 쓴다.

[0069] 따라서, 본 발명의 MapReduce 프로그래밍 모델에서는 사용자가 Update 함수를 작성하여 기존 결과를 얻어 오는 방법을 기술토록 하고, Reduce 함수가 호출될 때마다 Reduce 태스크 수행기 내부에서 해당 키에 해당하는 Update 함수를 수행하여 기존 결과값 old_values을 구한 후에 Reduce 함수의 입력으로 제공할 수 있다.

[0070] 이때, 사용자가 작성하는 Update 함수는 최종 결과가 파일이라면 파일에서 해당 키(key) 값의 현재까지의 결과를 읽어 올 수 있고, 최종 결과가 DB 테이블이라면 DB 테이블에서 해당 key 값에 해당하는 열(row)을 찾은 후 그 row의 값을 읽어 올 수 있다.

[0071] 도 5는 추가 입력 데이터가 수집되었는지를 확인하고 이를 처리하는 절차를 나타낸 흐름도이다.

[0072] 도 2 및 도 5를 참조하면, 스트림 데이터 모니터(112)는 주기적으로 입력 데이터 저장 위치(111)에 추가 수집 데이터가 있는지 확인할 수 있다(S501, S502).

[0073] 이때, 추가 수집 데이터가 없으면(No) 일정 주기로 수면을 취한 후(S503), 다시 추가 수집 여부를 확인한다(S501, S502).

[0074] 만일 추가 수집 데이터가 있으면(Yes), 이를 작업 관리기(102)에 통보한 후(S504) 일정 주기 동안 수면을 취한 후 다시 확인 작업을 반복한다.

[0075] 작업 관리기(102)는 추가 수집 데이터를 분석하여(S505), 데이터 개수 및 용량을 확인한 후, 해당 입력 데이터를 처리하기에 적정한 개수의 Map 태스크를 생성하고, 기존 Reduce 태스크와 같은 개수의 Reduce 태스크를 새로 생성한다(S506).

[0076] 생성된 Map 태스크는 작업 관리기(102)의 스케줄링에 의해 태스크 수행 노드의 Map 태스크 수행기에 할당되어 처리가 되며(S507), 생성된 Reduce 태스크는 작업 관리기(102)의 스케줄링에 의해 태스크 수행 노드의 Reduce 태스크 수행기에 할당되어 처리가 될 수 있다.

[0077] 생성된 Reduce 태스크에는 생성된 Map 태스크 수행 사실, 생성된 Map 태스크가 생성하게 될 중간 결과 위치 정보 및 기존 Reduce 태스크 최종 수행 결과 위치 정보 등을 제공될 수 있다(S508).

[0078] 생성된 Map 태스크가 수행이 완료되면(S509), 해당 Map 태스크가 생성한 중간 결과는 신규 Reduce 태스크에 복사되어 처리가 된다.

[0079] 또한, 사용자가 완료된 Map 태스크의 입력을 삭제하고자 하는 경우에(S510), 작업 관리기(102)는 입력 파일을

삭제하고 완료한다(S511).

[0080] 한편, 본 발명의 분산 병렬 처리 시스템은 새로운 스트림 데이터가 주기적으로 수집됨에 따라 여러 번의 Reduce 태스크가 새로 생성되어 이전 최종 결과를 참조하여 새로운 최종 결과를 만들게 되며, 이에 시간이 지남에 따라 출력 데이터 저장 위치에는 많은 버전, 즉 많은 개수의 최종 결과가 축적되게 된다.

[0081] 따라서, 도 1에 도시된 최종 결과 병합기(113)는 도 6에서 보는 바와 같은 절차로 기존 최종 결과를 병합하여 최종 결과의 개수를 줄이는 방법을 제공할 수 있다.

[0082] 도 6은 최종 결과를 병합하여 버전의 개수를 줄이는 방법을 나타낸 흐름도이다.

[0083] 도 1, 도 2 및 도 6을 참조하면, 최종 결과 병합기(113)는 최종 결과 병합이 시작되면(S601), 삭제 표시된 이전 버전의 최종 결과가 사용 중인지를 확인하고(S602), 사용중이 아니면(No) 이전 버전의 최종 결과를 삭제한다(S603).

[0084] 그러나, 확인 결과 이전 버전의 최종 결과가 사용중이라면(Yes), 출력 데이터 저장 위치(202) 내에 위치하는 최종 결과의 버전 수, 즉 최종 결과의 개수를 확인한다(S604).

[0085] 그리고, 확인된 버전 수를 설정값과 비교한다(S605). 여기서, 설정값은 사용자에 의해 설정된 최종 결과의 개수를 의미할 수 있다.

[0086] 비교 결과, 버전 수가 설정값보다 적으면(No), 일정 기간의 수면을 갖고(S606), 다시 최종 결과 병합 시작 단계(S601)를 수행한다.

[0087] 그러나, 비교 결과 버전 수가 설정값보다 많거나 같으면(Yes), 기존 버전의 최종 결과를 병합하여 하나의 새로운 최종 결과 버전을 생성한다(S607).

[0088] 이어, 병합 대상이 된 기존 버전의 최종 결과가 현재 수행 중인 Reduce 태스크에서 이전 최종 결과로서 사용되고 있는지를 확인한다(S608).

[0089] 확인 결과 사용중이 아니면 삭제하고(S609), 사용 중이면 삭제 표시를 한다(S610). 이후, 일정 기간의 수면 기간을 갖고(S606), 다시 최종 결과 병합 시작 단계(S601)를 수행한다.

[0090] 이상에서와 같이, 본 발명의 MapReduce 기반 분산 병렬 처리 시스템에서는 Reduce 태스크가 기본적으로 기존 결과가 존재한다는 가정으로 동작하기 때문에 하기와 같이 제한된 환경하에서 효율적인 분산 병렬 처리 기능을 제공할 수 있다.

[0091] 예컨대, 본 발명의 분산 병렬 처리 시스템은 1) Reduce 태스크의 개수가 최종 결과에 영향을 미치지 않거나, 2) Reduce 함수의 수행 결과가 Reduce 함수의 입력과 같은 키를 사용하거나, 3) 정확한 결과보다는 연속적으로 수집되는 스트림 데이터에 대한 실시간 분산 병렬 처리를 수행하고자 할 때, 강력한 분산 병렬 처리 기능을 제공할 수 있다.

[0092] 본 발명의 내용은 도면에 도시된 일 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 따라서 본 발명의 진정한 기술적 보호 범위는 첨부된 특허청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면의 간단한 설명

[0093] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 간단한 설명이 제공된다.

[0094] 도 1은 본 발명에 따른 분산 병렬 처리 시스템의 시스템 구성도이다.

[0095] 도 2는 본 발명에 따른 분산 병렬 처리 방법의 동작을 나타낸 예시도이다.

[0096] 도 3은 출력 데이터 저장 위치에 최종 결과를 생성하는 디렉토리 구성을 나타낸 예시도이다.

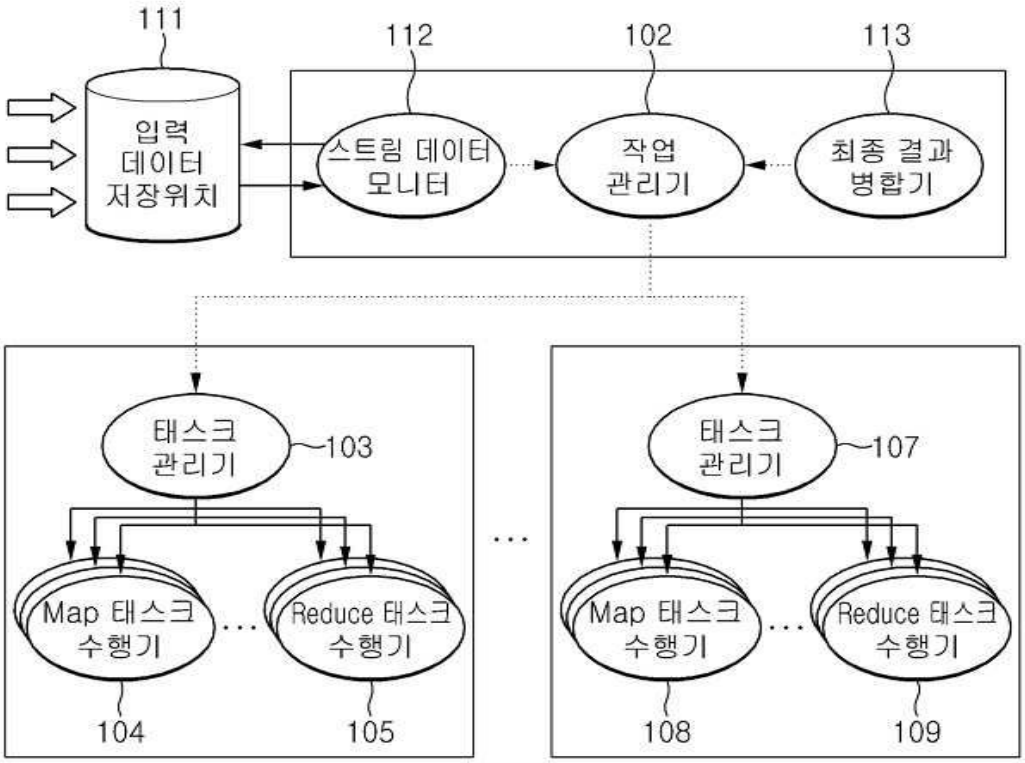
[0097] 도 4는 본 발명의 MapReduce 프로그래밍 모델의 예시도이다.

[0098] 도 5는 추가 입력 데이터가 수집되었는지를 확인하고 이를 처리하는 절차를 나타낸 흐름도이다.

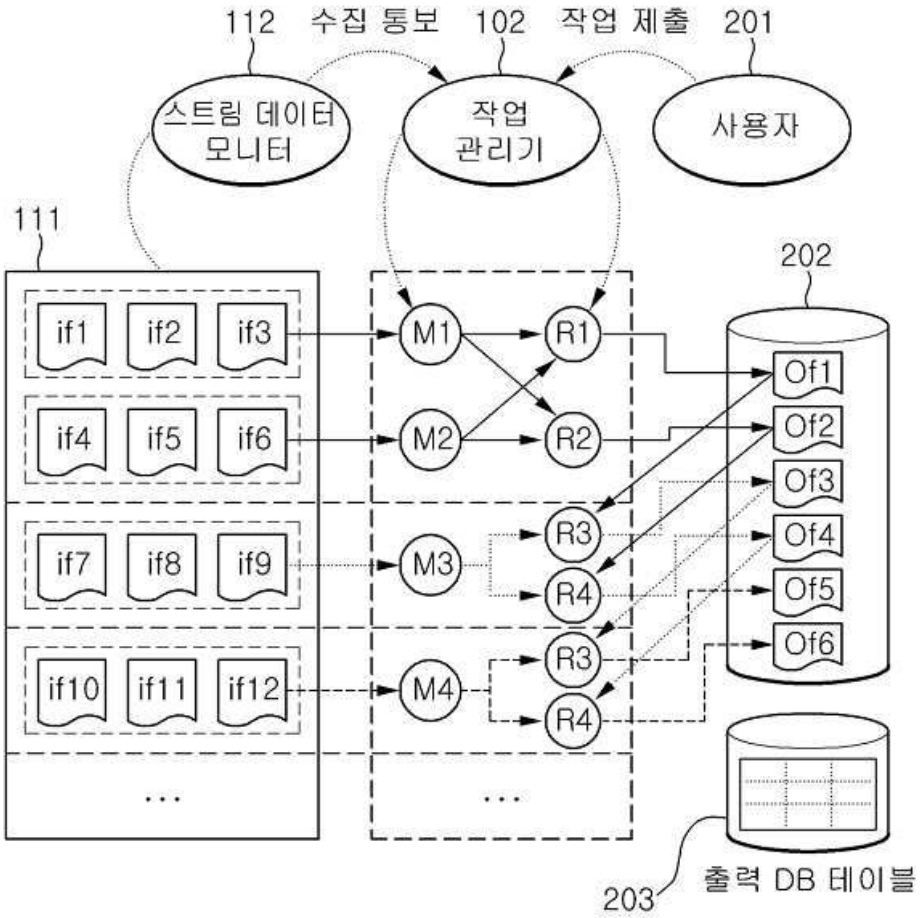
[0099] 도 6은 기존 최종 결과를 병합하여 버전의 개수를 줄이는 방법을 나타낸 흐름도이다.

도면

도면1



도면2



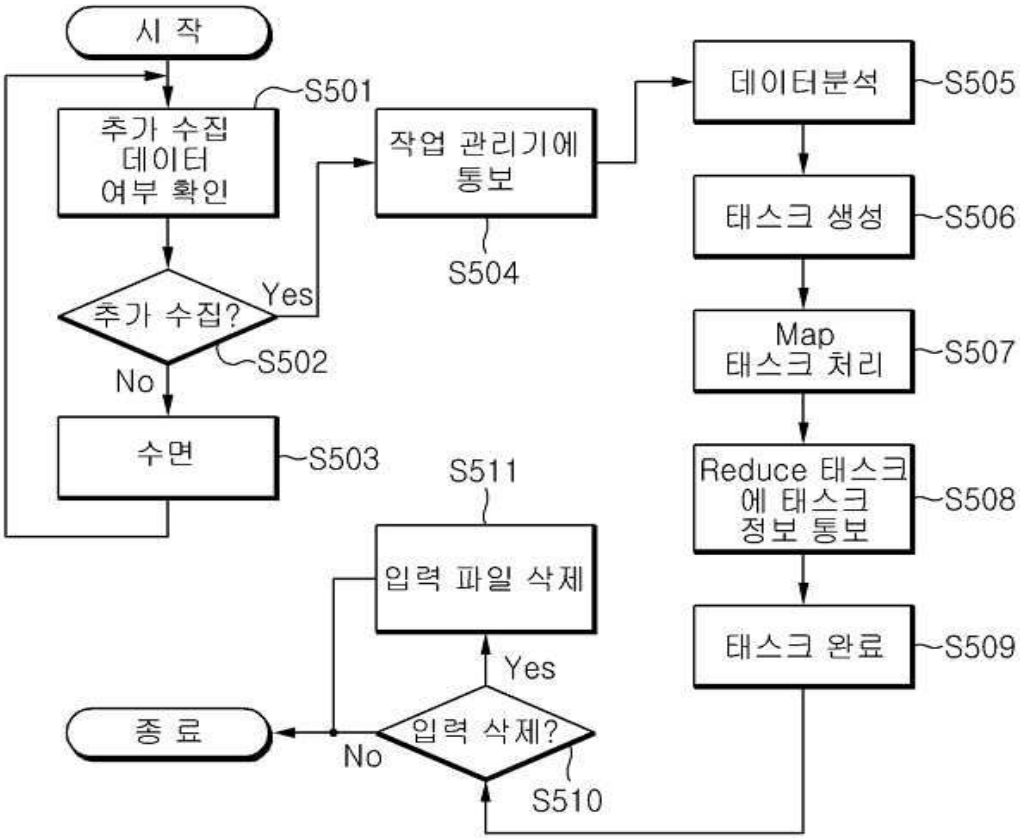
도면3

+ output_dir
  + 1254293251990
    + r1
    + r2
+ output_dir
  + 1254293251991
    + r3
    + r4
+ output_dir
  + 1254293251992
    + r5
    + r6
  . . .

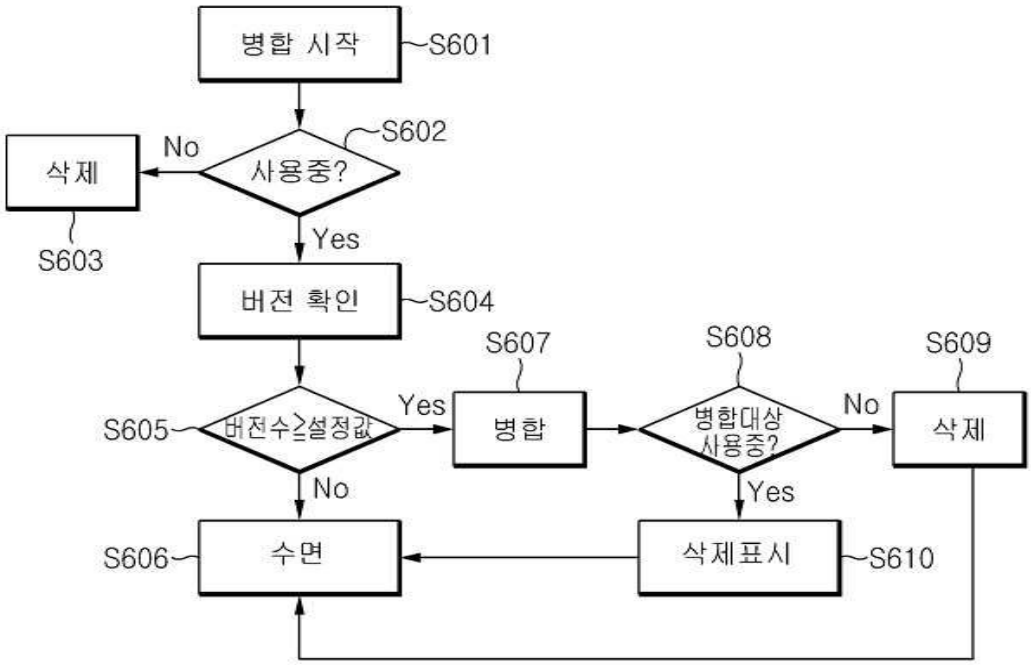
도면4

```
401 map(String key, String value):
      // key: document name
      // value: document contents
      for each word w in value:
        EmitIntermediate(w, "1");

                                                          404
402 reduce(String key, Iterator new_values, Iterator old_values):
      // key: a word
      // new_values: a list of counts from new data
      // old_values: a list of counts from old data
      int result = 0;
      for each v in old_values:
        result += ParseInt(v);
      for each v in new_values:
        result += ParseInt(v);
      Emit(AsString(result));

403 update(String key):
      // key: a word
      // old_values: a list of counts from old data
      old_values = select_from_db(key);
      // Or, old_values = select_from_file(key);
      Emit(old_values);
```

도면5



도면6


병합 시작
S601
S602
사용중?
No
삭제
S603
Yes
버전 확인
S604
S605
버전수≥설정값
Yes
S607
병합
S608
병합대상 사용중?
No
S609
삭제
No
Yes
S606
수면
삭제표시
S610