특집 03

빅 데이터에 대한 맵리듀스 기반의 빈발 패턴 마이닝 기술

전강욱 · 김민수 (DGIST)

목 차 »

1. 빅 데이터 빈발 패턴 마이닝
2. 맵리듀스 프레임워크
3. 맵리듀스 기반 빈발 패턴 마이닝 알고리즘
4. 맵리듀스 기반 빈발 패턴 마이닝 알고리즘 성능 분석
5. 결 론

1. 빅 데이터 빈발 패턴 마이닝

최근 IT 기술의 발전과 웹, SNS 등의 확산 등의 이유로 생성되거나 저장되는 데이터의 크기가 급격하게 증가하고 있다. 그리고, 이러한 빅 데이터를 분석함으로써 새로운 사실과 가치 있는 직관(insight)을 발견하려는 빅 데이터 마이닝(big data mining) 기술이 중요해지고 있다. 데이터 마이닝은 “대용량의 데이터로부터 직접적으로 표출되지 않은 유용한 지식을 찾아내는 일련의 과정”이라 정의되고, 90년대 초반부터 지식 발견(Knowledge Discovery in Database), 정보 발견(Information Discovery) 등의 이름으로 소개되었다. 데이터 마이닝의 주요 기법들은 빈발 패턴 마이닝(frequent pattern mining), 분류(classification), 군집화(clustering)등이 있다.

데이터 마이닝 기법들 중 빈발 패턴 마이닝은 데이터베이스의 트랜잭션(transaction)등의 데이터에서 빈번하게 발생하는 아이템들의 집합을 찾아내는 마이닝 기법이다[1]. 빈발 패턴 마이닝은 다음과 같이 정의된다. $I=\{i_1,i_2,\cdots,i_m\}$를 아이템들의 집합으로 정의하고, 데이터베이스 D는 트랜잭션들의 집합으로 정의한다. 각 트랜잭션 T는 $T\subset I$인 아이템 집합이다. 아이템 집합 X가 $X\subset T$일 경우, “트랜잭션 T가 아이템 집합 X를 포함 한다”라고 한다. D에 속하는 s%의 트랜잭션들이 아이템 집합 X를 포함한다면, 우리는 X의 지지도를 s라고 한다. 또한 우리는 미리 정의된 최소 지지도(minimum support) 이상을 갖는 아이템 집합을 빈발 아이템 집합이라 한다. 그리고 k개의 아이템들로 이루어진 빈발 아이템 집합을 빈발 k-아이템 집합(frequent k-itemset)이라 한다. 빈발 패턴 마이닝 문제는 사용자가 지정한 최소 지지도 이상의 지지도를 갖는 모든 빈발 아이템 집합들을 구하는 것이다. 기존에 널리 사용되는 빈발 패턴 마이닝 알고리즘들은 Apriori 방법[2],

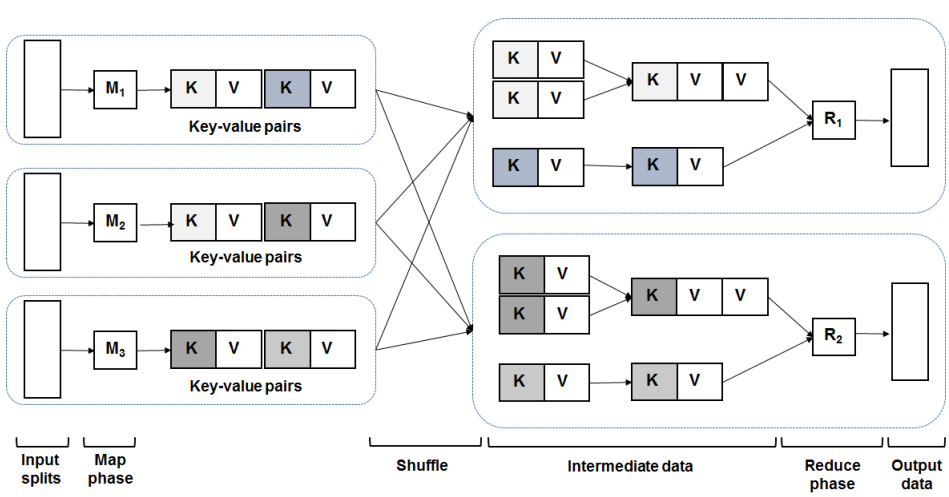
FP-Growth 방법[3], Partition 방법[4], 그리고 Vertical 방법[5] 등이 있다.

기존에 제안된 빈발 패턴 마이닝 알고리즘들은 대부분 단일 컴퓨터상에서의 메모리 기반 알고리즘들로써 입력 데이터의 크기가 작거나 최소 지지도가 높은 경우에는 적정 시간 내에 빈발 패턴들을 구할 수 있다. 그러나, 입력 데이터의 크기가 너무 커지거나, 최소 지지도가 너무 낮은 경우에는 메모리 등의 문제로 빈발 패턴들을 구하지 못하거나, 또는 구하더라도 시간이 너무 오래 걸리는 문제를 가지고 있다[6-10].

앞서 언급한 기존 알고리즘들의 문제를 해결하기 위하여 최근 여러 대의 컴퓨터들로 구성된 클러스터 상에서 맵리듀스[11] 프레임워크 기반으로 빈발 패턴 마이닝을 수행하는 알고리즘에 관한 연구가 이루어지기 시작했다. 본 논문에서는 단일 컴퓨터 기반의 빈발 패턴 마이닝 알고리즘들이 가진 한계를 극복하기 위해 최근 제안된 맵리듀스 기반의 주요한 빈발 패턴 마이닝 알고리즘들을 설명하고, 그들 사이의 성능 비교를 수행하는 것을 목적으로 한다. 이 연구는 본 저자들이 아는 범위에서 맵리듀스 기반의 빈발 패턴 마이닝 알고리즘들에 대한 최초의 성능 비교 연구이다.

2. 맵리듀스 프레임워크

맵리듀스 프레임워크는 대용량 데이터를 분산 처리하기 위한 목적으로 개발된 프로그래밍 모델이다. 맵리듀스 프레임워크는 맵(map)과 리듀스(reduce)로 이루어진 비교적 간단한 인터페이스, 분산 파일 시스템의 데이터 복제(replication)에 기반을 둔 내고장성(fault-tolerant), 높은 확장성(scalability) 등의 장점을 가지고 있어 빅 데이터 분석에 있어서 사실상의 표준(de facto)으로 자리 잡았다. (그림 1)과 같이 맵리듀스의 데이터 처리 과정은 맵 단계(map phase)와 리듀스 단계(reduce phase)로 나누어진다. 각 단계의 입출력은 모두 키-값 쌍(key-value pair)의 형태를 따른다. 맵 단계의 각 맵퍼(mapper)는 하나의 스플릿(split)을 입력으로 받고, 사용자가 정의한 맵 함수를 수행한 후에 키-값 쌍으로 이루어진 중간 데이터를 만든다. 맵 단계를 거친 중간 결과들은 키의 순서대로 정렬 및 그룹핑(grouping) 된다. 그리고 정렬된 중간 결과들은 키 값에 따라 특정 리듀서로 전



(그림 9) 맵리듀스의 데이터 처리 과정

송된다. 리듀스 단계에서 각 리듀서는 각 키에 대해 값들의 리스트를 입력으로 받아 리스트 전체를 순환하며 사용자가 정의한 리듀스 함수를 수행한다.

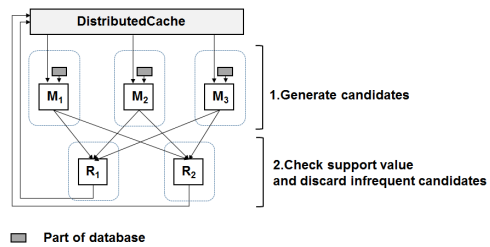
맵리듀스 프레임워크의 분산 캐시(DistributedCache)는 특정 파일을 복사하여 모든 노드들에 분배(broadcast)하는 기능을 수행한다[12]. 일반적으로 분산 캐시는 비공유(shared-nothing) 구조인 맵리듀스에서 데이터를 공유하기 위하여 유용하게 사용될 수 있다[6-9,12]. 예를 들어 분배 조인(broadcast join) 연산은 테이블 R과 테이블 L이 |R| << |L|의 관계를 가질 경우 테이블 R과 테이블 L을 네트워크를 통해 재분배하는 대신, 네트워크 오버헤드(network overhead)를 줄이기 위하여 테이블 R을 분산 캐시를 이용하여 모든 노드들에 분배(broadcast)시키고 테이블 L을 모든 노드들에 파티션하여 저장한 후, 각 노드에서 조인 연산을 수행하는 방법을 사용한다.

3. 맵리듀스 기반 빈발 패턴 마이닝 알고리즘

본 장에서는 맵리듀스 기반의 주요한 빈발 패턴 마이닝 알고리즘들에 대해 설명을 한다. 설명에서 사용되는 그림들(그림 2~5)에서 M은 하나의 맵을 나타내고, R은 하나의 리듀스를 나타낸다.

3.1 Apriori 기반 알고리즘

맵리듀스에 기반을 둔 Apriori 알고리즘[6]은 기존의 단일 컴퓨터 기반의 Apriori와 같이 반복적인 접근 방법을 통하여 빈발 패턴들을 구한다. 해당알고리즘은 (그림 2)와 같이 이전 반복 단계에서 얻은 (k-1)-빈발 아이템 집합들을 분산 캐시를 통하여 모든 노드들에 분배한다. 그리고 각 맵퍼

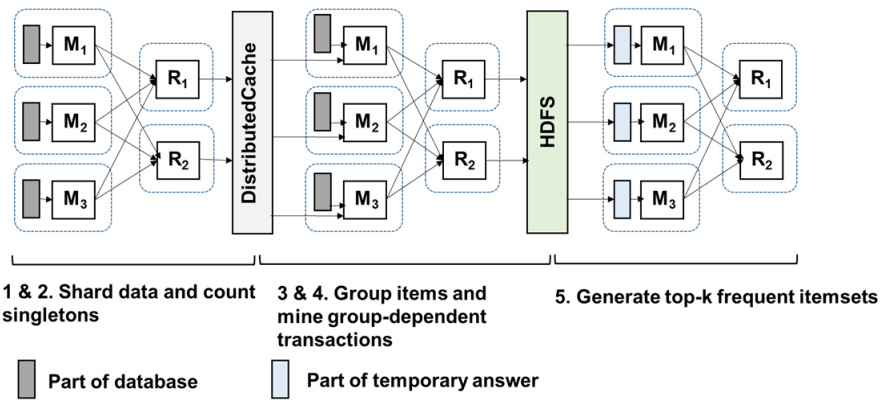


(그림 2) 맵리듀스를 기반으로 둔 Apriori 알고리즘

는 (k-1)-빈발 아이템 집합들을 이용하여 k-빈발 아이템 집합 후보들을 생성한다. 생성된 k-빈발 아이템 집합 후보 A는 데이터베이스 D에 속하는 모든 트랜잭션들과 비교된다. 즉, A가 B∈D인 트랜잭션 B에 포함되면 A와 B에서의 A의 지지도 1은 키와 값으로써 출력된다. 리듀스 단계에서 리듀서는 입력으로 받은 값 리스트를 하나의 값으로 취합하고, 이 값이 사용자가 정의한 최소 지지도보다 크거나 같으면 입력 키는 빈발 k-아이템 집합으로써 출력된다. 이와 같은 맵리듀스 작업은 빈발 아이템 집합들이 모두 구해질 때까지 반복된다.

3.2 FP-Growth 기반 알고리즘

Parallel FP-Growth[7]은 상위 k개(top-k)의 빈발 아이템 집합들을 구하고자 맵리듀스를 이용하여 FP-Growth를 병렬화한 알고리즘이다. Parallel FP-Growth는 독립적인 마이닝 연산을 위한 전처리 단계, 마이닝 단계, 취합 단계의 세 단계로 구성된다. (그림 3)은 Parallel FP-Growth의 전체적인 과정을 보여준다. 전처리 단계는 하나의 맵리듀스 작업을 통하여 사용자가 지정한 최소 지지도 이상의 지지도를 갖는 빈발 아이템들을 찾아낸다. 그리고 빈발 아이템들은 사용자가 입력한 값에 따라 여러 개의 그룹들로 나뉘는데, 이 때 각 그룹은 그룹 식별자를 갖는다. 전처리 단계에서 얻어진 빈발 아이템들과 그룹들은 분산 캐시



(그림 3) Parallel FP-Growth 알고리즘

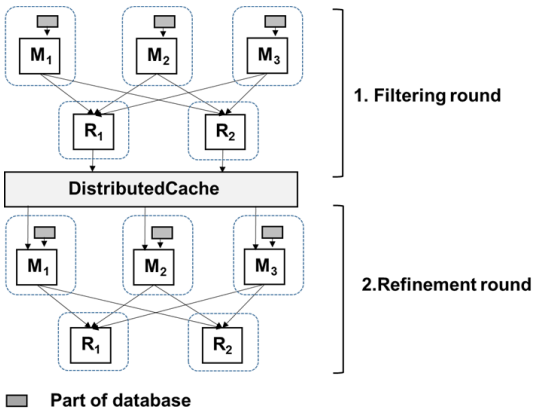
를 통해 마이닝 단계를 시작하기 전에 모든 노드들에게 분배된다. 마이닝 단계는 하나의 맵리듀스 작업을 통해서 수행된다. 마이닝 단계의 각 맵퍼는 그룹 의존적인 데이터베이스(group-dependent database)들을 만들기 위하여 레코드의 모든 프리픽스(prefix)들의 마지막 아이템을 기준으로 그룹 식별자와 관련된 아이템 집합을 키와 값으로써 출력한다. 리듀스 단계에서 각 리듀서는 같은 그룹 식별자를 갖는 그룹 의존적인 트랜잭션(group-dependent transaction)들을 입력으로 받은 후에 단일 컴퓨터 기반의 FP-Growth를 이용하여 빈발 아이템 집합들을 구한다. 취합 단계는 각 아이템 별 상위 k개의 빈발 아이템 집합들을 구한다.

3.3 파티션(partition) 기반 알고리즘

파티션 기반 알고리즘[9,10]에 관한 연구들은 트랜잭션 데이터[9]나 그래프 데이터[10]에 대해서 연구가 된 바 있다. 두 가지 데이터에 관련한 파티션 기반 알고리즘들에 관한 연구는 다루는 데이터의 형태와 각 데이터 형태에 따른 최적화 방법만 다르고, 기본적으로 알고리즘을 수행하는 방식은 같다. 따라서 본 논문에서는 트랜잭션 데이터에 관련한 알고리즘만을 다룬다.

파티션 기반 알고리즘은 입력 데이터에 대해서 기존의 파티션 알고리즘과 같이 데이터베이스를 몇 개의 파티션들로 나눈다. 파티션 P는 데이터베이스 D와 $P \subset D$의 관계를 갖는다. 그리고 파티션들끼리는 서로 겹치는 부분이 없다(i와 j가 같지 않을 경우 $P_i \cap P_j = \varnothing$ 이다). 지역 지지도(local support)는 임의의 파티션 P에서의 임의의 아이템 집합 X의 지지도이다. 즉, P에 속해있는 s%의 트랜잭션들이 X를 포함한다면, P에서의 X의 지역 지지도는 s이다. 또한, 임의의 파티션 P에서 아이템 집합 X의 지역 지지도가 사용자가 정의한 최소 지지도보다 크거나 같으면 아이템 집합 X는 파티션 P에서 지역적으로 빈발한 아이템 집합이라 정의된다. 파티션 기반 알고리즘은 전체 데이터베이스 D에서 빈발한 아이템 집합들을 찾기 위하여 "임의의 아이템 집합 X가 전역적으로 빈발하려면 반드시 적어도 하나의 파티션에서 지역적으로 빈발 하여야한다"라는 정리를 사용한다.

(그림 4)와 같이 파티션 기반 알고리즘은 여과(filtering) 단계와 정제(refinement) 단계로 구성되고 각각의 단계는 하나의 맵리듀스 작업이다. 파티션 기반 알고리즘에서 파티션들은 맵리듀스 프레임워크에 의해 자동으로 생성된다. 여과 단계



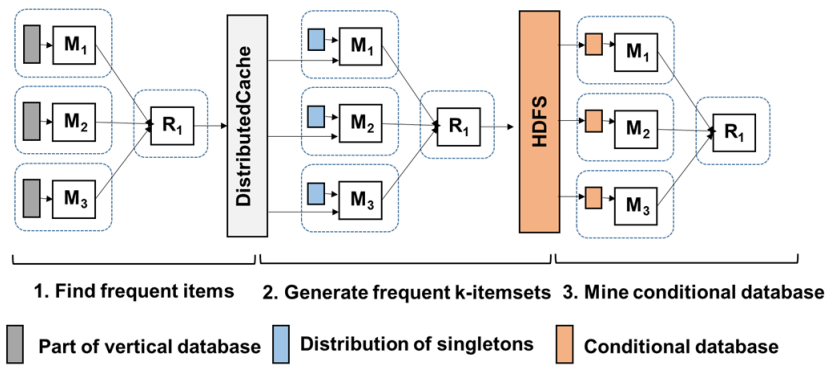
(그림 4) 맵리듀스를 기반으로 둔 파티션 알고리즘

는 지역적으로 빈발한 아이템 집합들을 구한다. 여과 단계의 각 맵퍼는 파티션을 입력으로 받아 FP-Growth나 Apriori 알고리즘과 같은 단일 컴퓨터 기반 빈발 패턴 마이닝 알고리즘을 통해 지역적으로 빈발한 아이템 집합들을 구하고 리듀스 단계는 여러 개의 맵퍼에서 출력한 동일 빈발 아이템 집합들에 대하여 한번만 출력함으로써 불필요한 중복을 제거한다. 위의 작업의 결과는 거짓 양성(false positive)들을 포함하는 문제점을 갖는다. 따라서 해당 알고리즘은 정제 단계를 통해서 거짓양성들을 제거한다. 거짓양성들을 제거하기 위해서 파티션 기반 알고리즘은 분산 캐시를 통해 이전 단계에서 얻은 결과를 모든 노드들에게 분배한다. 즉, 정제 단계의 각 맵퍼는 여과 단계의 작업의 결과 C와 임의의 파티션 P를 입력 받고, C에 속하는 모든 아이템 집합들의 파티션 P에서의 지지도를 구하여 아이템 집합들과 아이템 집합들의 지역 지지도를 키와 값으로써 출력한다. 정제 단계의 각 리듀서는 아이템 집합과 해당 아이템 집합의 각 파티션에서의 지지도 값들로 구성된 리스트를 키와 값으로써 입력받아 리스트에 속하는 모든 값들을 더하여 하나의 값으로 취합한다. 그 후에 리듀서는 취합된 값이 사용자가 입력한 최소 지지도보다 크거나 같으면 키를 빈발 아이템 집합으로써 출력한다.

3.4 Vertical 기반 알고리즘

Vertical 기반 알고리즘은 데이터를 나누는 것보다 탐색 공간(search space)을 나누는 방법을 사용한다. [8]은 Vertical 방법에 기반을 둔 두 가지 알고리즘들을 제안하였다. 하나는 Dist-Eclat이고, 다른 하나는 BigFim이다. 두 알고리즘 모두 데이터베이스를 여러 개의 조건부 데이터베이스(conditional database)들로 나눔으로써 탐색 공간을 나눈다. 이때 두 알고리즘 모두 데이터베이스에 속하는 트랜잭션들이 임의의 빈발 k-아이템 집합을 포함하는지의 여부에 따라 데이터베이스를 여러 개의 조건부 데이터베이스들로 나눈 후에 각 맵퍼에 조건부 데이터베이스를 분배한다. 이때 조건부 데이터베이스들은 단일 컴퓨터 기반의 Eclat에서 사용되는 프리픽스 트리(prefix tree)이다. 그리고 각 맵퍼는 네트워크 오버헤드 없이 개별적으로 마이닝 연산을 함으로써 빈발 아이템 집합들을 구한다.

(그림 5)는 Dist-Eclat의 작업 흐름도이다. 해당 알고리즘은 세 단계로 나누어지는데, 첫 번째 단계는 하나의 맵리듀스 작업을 통하여 빈발 아이템들을 구한다. 이 때 첫 번째 단계는 Vertical 포맷의 데이터베이스를 입력으로 받아 빈발 아이템들과 해당 아이템들의 트랜잭션 식별자 리스트(tid-list)를 단일 컴퓨터에 기반을 둔 Eclat을 이용해 찾아낸 후, 키와 값으로써 출력한다. 두 번째 단계는 빈발한 k-아이템 집합들을 Eclat 알고리즘을 이용하여 구한 뒤 조건부 데이터베이스들을 생성한다. 세 번째 단계에서 각 맵퍼는 조건부 데이터베이스를 입력 받아 Eclat을 통하여 다른 맵퍼와 통신 없이 개별적으로 마이닝 연산을 한다.



(그림 5) Dist-Eclat 알고리즘

하지만 Dist-Eclat은 데이터의 크기가 큰 경우 첫 번째 단계에서 아이템들의 트랜잭션 식별자 리스트를 메모리에 유지하지 못하여 마이닝 연산을 할 수 없는 경우가 있다[8]. 따라서 [8]은 Dist-Eclat이 갖는 메모리 문제를 해결하기 위하여 BigFim을 제안하였다. BigFim은 Apriori 방법과 Vertical 방법의 하이브리드(hybrid) 방법이다. 이 알고리즘은 맵리듀스에 기반을 둔 Apriori와 같은 방법으로 빈발 k-아이템 집합을 구함으로써 메모리 문제를 해결한다. BigFim은 조건부 데이터베이스를 만들 때 사용되는 빈발 k-아이템 집합들을 구하는 방식을 제외하고 Dist-Eclat과 동일한 방법으로 마이닝 연산을 수행한다.

4. 맵리듀스 기반 빈발 패턴 마이닝 알고리즘의 성능 분석

본 장은 제 3장에서 설명한 맵리듀스 기반의 빈발 패턴 마이닝 알고리즘들의 성능을 비교한다. 이때, [7]은 일반적인 빈발 패턴 마이닝 알고리즘이 아니라 상위 k개의 빈발 패턴들을 구하는 마이닝 알고리즘이기 때문에 다른 알고리즘들과 공정한 성능 비교가 어렵다. 따라서 본 논문은 [7]을 실험 대상에서 제외한다. 또한 [8]에서 제안한 알고리즘들 중 Dist-Eclat은 크기가 큰 데이터를 다룰 경우 3.4절에서 언급한 메모리 문제가 발생하기 때문에 실험 대상에서 제외한다. 나머지 비교 대상 알고리즘들의 성능을 비교하기 위해 여러 종류의 데이터에 대해 최소 지지도를 변화시키면서 실행 시간을 측정한다.

4.1 실험 데이터 및 실험 환경

실험 데이터로는 세 가지 종류의 실제 데이터를 사용한다. <표 1>은 실험에 사용된 실제 데이터들의 특성을 보여준다. 실제 데이터는 Webdocs 데이터, Mushroom 데이터, Retail 데이터이고, 해당 데이터들은 [13]에서 얻은 데이터로써 데이터 마이닝 분야에서 성능 평가를 위해 주로 많이 사용되는 데이터이다. 이때, Webdocs 데이터의 경우 최대 트랜잭션 길이만 알 수 있고, 평균 트랜잭션 길이는 알기 어려워, 최대 트랜잭션 길이를 명시한다.

실험을 위해 20 대의 컴퓨터들로 이루어진 클러스터를 사용한다. 각 노드는 CentOS 6.4 운영체제, 인텔 제온 2.60GHz 8코어 CPU 두 개와 64GB 크기의 메모리, 그리고 6TB 디스크로 구성되어있고, 각 노드들은 1Gbps 네트워크로 서로

〈표 1〉 실제 데이터 특징

Dataset name	# Trans	# Items	Avg trans. size
Webdocs	1,692,082	5,267,656	71.472 (Max trans. size)
Mushroom	8,124	119	23
Retail	88,162	16,470	10.3

연결된다. 그리고, 맵리듀스 프레임워크로써 Hadoop 버전 1.2.1[14]을 사용한다.

4.2 최소 지지도에 따른 확장성

4.2.1 Retail 데이터에 대한 성능 비교

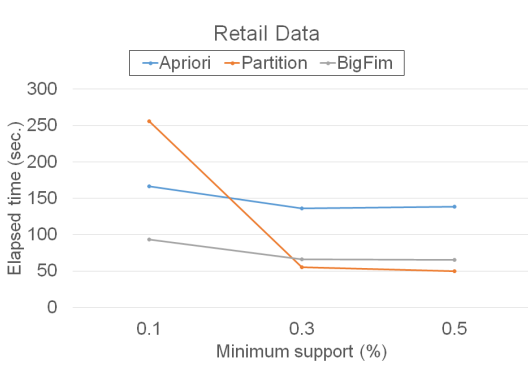
(그림 6)은 희소(sparse)한 데이터인 Retail에 대하여 최소 지지도 값이 변화할 때의 맵리듀스 기반 빈발 패턴 마이닝 알고리즘들의 실행 시간을 나타낸다. Apriori 기반 알고리즘과 BigFim 알고리즘은 최소 지지도가 감소함에 따라서 선형적으로 실행 시간이 증가함을 알 수 있다. 하지만, 파티션 기반 알고리즘은 최소 지지도가 0.1인 경우에 실행 시간이 가파르게 증가한다. 그 이유는 최소 지지도가 낮아짐에 따라서 여과 단계에서 생성되는 빈발 아이템 집합 후보들이 많아지기 때문이다.

특이 사항으로 최소 지지도가 높은 경우에 Apriori 기반 알고리즘의 실행 시간이 가장 크다. 그 이유는 Apriori 기반 알고리즘에서 반복적으로 맵리듀스 작업들을 실행함에 있어 발생하는 입력 데이터를 반복적으로 읽는 것과 같은 불필요한 비용 때문이다. 그리고 BigFim은 최소 지지도가 낮아지더라도 다른 알고리즘들과 비교하여 실행 시간의 증가폭이 적다. 이는 해당 알고리즘이 조건부 데이터베이스를 만듦으로써 마이닝 연산을 독립적으로 수행하기 때문이다.

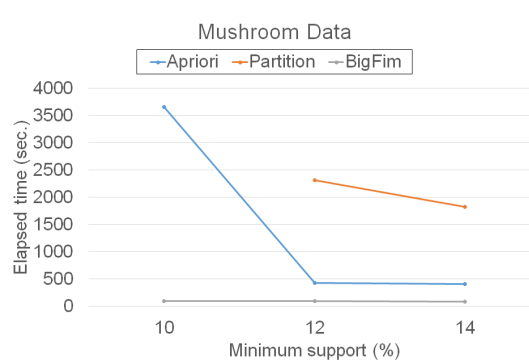
4.2.2 Mushroom 데이터에 대한 성능 비교

(그림 7)은 밀집(dense)한 데이터인 Mushroom에 대하여 최소 지지도 값이 변화할 때의 맵리듀스 기반 빈발 패턴 마이닝 알고리즘들의 실행 시간을 나타낸다. 파티션 알고리즘은 최소 지지도가 10인 경우에 실행 시간이 너무 오래 걸리기 때문에 해당 그림에 포함시키지 않았다.

(그림 7)에서 파티션 알고리즘은 최소 지지도에 상관없이 성능이 가장 나쁘다. 그 이유는 밀집한 데이터를 파티션하여 노드 하나가 다루는 데이터의 양을 줄였다 하더라도 각 파티션에서 생성되는 빈발 아이템 집합 후보들의 수가 많아 정제 단계에서 걸리는 시간이 크기 때문이다. Apriori 기반 알고리즘은 최소 지지도가 10인 경우에 실행 시간이 가파르게 증가하였다. 이는 최



(그림 6) Retail 데이터에 대한 맵리듀스 기반 빈발 패턴 마이닝 알고리즘의 실행 시간

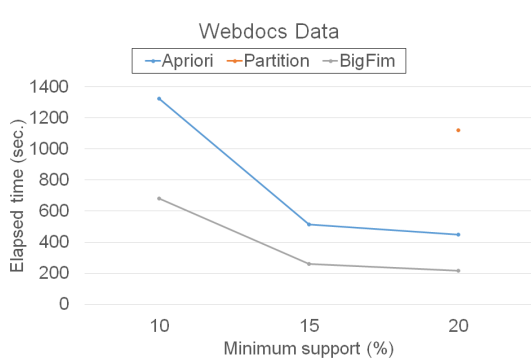


(그림 7) Mushroom 데이터에 대한 맵리듀스 기반 빈발 패턴 마이닝 알고리즘의 실행 시간

소 지지도가 낮아짐에 따라서 Mushroom 데이터에서 생성되는 패턴들의 길이가 길어져 맵리듀스 작업의 수가 증가하기 때문이다. 또한 (그림 7)을 통해 BigFim 알고리즘은 최소 지지도와 관계없이 가장 좋은 성능을 갖는 것을 알 수 있다. 이는 Mushroom 데이터가 포함하는 트랜잭션의 수가 적어 조건부 데이터베이스를 만드는 비용이 거의 없고, 해당 알고리즘의 마이닝 단계가 독립적으로 이루어지기 때문이다.

4.2.3 Webdocs 데이터에 대한 성능 비교

(그림 8)은 맵리듀스 기반 알고리즘들이 Webdocs 데이터의 패턴들을 찾을 때 걸리는 실행 시간을 나타낸다. Webdocs 데이터는 Mushroom 데이터와 마찬가지로 밀집 데이터이고, 우리가 구할 수 있는 실제 데이터 중에서 가장 큰 데이터 중 하나이다[15]. (그림 8)에서 파티션 기반 알고리즘은 실행 시간이 너무 오래 걸리기 때문에 최소 지지도가 20일 경우만 측정하였다. 앞서 수행한 실험들과 마찬가지로 파티션 알고리즘은 여과 단계에서 생성되는 빈발 아이템 집합들이 너무 많아 정제 단계에서 실행 시간이 길다. 또한, Apriori 기반 알고리즘은 Webdocs 데이터에서 나타나는 긴 패턴으로 인한 많은 수의 맵리듀스의 작업 때문에 최소 지지도가 10인 경우 실행 시간이 급격하게 증가한다. BigFim 알고리즘의 경우에도 최소 지지도가 10인 경우 Apriori 기반 알고리즘과 유사한 경향을 보인다. 이는 Webdocs 데이터가 포함하는 많은 수의 트랜잭션 때문에 조건부 데이터베이스를 만드는데 많은 비용이 수반되기 때문이다.



(그림 8) Webdocs 데이터에 대한 맵리듀스 기반 빈발 패턴 마이닝 알고리즘의 실행 시간

5. 결 론

본 논문은 맵리듀스를 기반으로 둔 빈발 패턴 마이닝 알고리즘들에 대하여 기술하고, 해당 알고리즘들의 성능을 동일한 환경에서 분석하였다. 최소 지지도에 따른 확장성에 관한 실험에서 BigFim 알고리즘은 대부분의 실험 결과에서 가장 좋은 성능을 갖는다. 특히, 실험에 사용된 데이터가 Webdocs, Mushroom 데이터와 같이 밀집한 성질을 갖는 경우, 해당 알고리즘은 다른 알고리즘과 다르게 적정시간 내에 모든 빈발 패턴들을 구한다. Apriori 방법 기반 알고리즘과 파티션 기반 알고리즘은 실험에 사용된 데이터가 Retail 데이터와 같이 희소 데이터인 경우에는 좋은 성능을 갖지만, 밀집 데이터를 처리할 경우에 심각한 성능 저하를 갖는다.

참 고 문 헌

[1] R. Agrawal, T. Imieli´nski, and A. Swami, “Mining association rules between sets of items in large databases”, In SIGMOD ’93, pages 207–216. ACM, 1993.

[2] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules in large databases”, In Proc. VLDB Endow., VLDB ’94, pages 487–499, San Francisco, CA, USA, 1994.

[3] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation”, In SIGMOD ’00, pages 1–12, ACM, 2000.
[4] A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association rules in large databases", in Proceedings of the 21st VLDB Conference Zurich, Swizerland, 1995.
[5] M. J. Zaki and K. Gouda, “Fast vertical mining using diffsets”, In Proc. ACM SIGKDD, pages 326–335, 2003.
[6] M.-Y. Lin, P.-Y. Lee, and S.-C. Hsueh, “Apriori-based frequent itemset mining algorithms on mapreduce”, In ICUIMC ’12, pages 76:1–76:8. ACM, 2012.
[7] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Chang, “Pfp: parallel fpgrowth for query recommendation”, In RecSys ’08, pages 107–114. ACM, 2008.
[8] S. Moens, E. Aksehirli, and B. Goethals, "Frequent Itemset Mining for Big Data", Big Data, 2013 IEEE International Conference on, pages 111-118, 2013.
[9] T. Xiao, C. Yuan, and Y. Huang, “PSON: A Parallelized SON Algorithm with MapReduce for Mining Frequent Sets”, In Proceedings of the 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming (PAAP '11). IEEE Computer Society, Washington, DC, USA, 252-257. 2011.
[10] W. Lin, X. Xiao, and G. Ghinita, “Large-Scale Frequent Subgraph Mining in MapReduce”, Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE), pages 844-855, 2014.
[11] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters”, Commun. ACM, 51(1):107–113, 2008.
[12] T. White, Hadoop: The Definitive Guide. O’Reilly Media, Inc., 2009.
[13] FIMI (Frequent Itemset Mining Implementations Repository). http://fimi.cs.helsinki.fi.
[14] Apache Hadoop. http://hadoop.apache.org.
[15] G.-P. Chen, Y.-B. Yang, and Y. Zhang, “Mapreduce-based balanced mining for closed frequent itemset”, in IEEE 19th International Conference on Web Service, pp. 652–653, 2012.

저 자 약 력

전 강 욱

이메일 : kw.chon@dgist.ac.kr

- 2012년~현재 DGIST 정보통신융합공학전공 석·박사 통합과정
- 2012년 전북대학교 컴퓨터공학과(학사)
- 관심분야 : 데이터베이스, 데이터마이닝, 분산 처리 시스템

김 민 수

이메일 : mskim@dgist.ac.kr

- 2011년~현재 DGIST 정보통신융합공학전공 조교수
- 2009년~2011년 미국 IBM Almaden Research Center 연구원
- 2007년~2009년 미국 UIUC 포스닥 연구원
- 2006년 KAIST 전산학과 박사
- 관심분야 : 빅 데이터 시스템, 빅 데이터 마이닝, 바이오인포매틱스