2002 년 한국정보처리학회 춘계학술발표논문집 제 9 권 제 1 호

병렬 연관 규칙 마이닝을 위한 동적 부하 분산 알고리즘

김민호, 김계형, R.S. Ramakrishna
광주과학기술원 정보통신공학과
e-mail : mhkim@kjist.ac.kr

Dynamic Load Balancing Algorithm for Parallel Association Rule Mining

Minho Kim, Gye Hyung Kim, R.S. Ramakrishna
Dept. of Information and Communications, K-JIST

요 약

본 논문에서는 대표적인 데이터 마이닝 기법 중 하나인 연관 규칙 마이닝에 대해 PC 성능의 급속한 발전으로 인한 PC 클러스터 시스템의 이종화의 필연성을 효과적으로 대처할 수 있는 부하 분산 알고리즘을 제안한다. 제안한 부하 분산 기법은 실행 전 성능을 미리 측정할 필요가 없이 실행 중에 성능을 측정할 수 있는 동적 부하 분산 알고리즘으로써 노드들 사이에 성능 정보의 교환 비용밖에 요구되지 않는다. 실험 결과는 제안한 알고리즘이 이종의 클러스터 시스템의 효율성을 극대화함을 보여준다. 또한 본 논문에서는 부하 분산 알고리즘의 성능을 분석할 수 있는 방법을 제시한다.

1. 서론

데이터 마이닝이란 대량의 데이터로부터 이전에는 알려지지 않았던 유용한 정보 (패턴 또는 지식으로도 불리워짐)를 효과적으로 추출하는 과정을 의미한다.

최근 인터넷의 발달과 전자 시장의 활성화는 더욱 더 많은 데이터의 생성 및 저장을 초래하고 있으며 과학 분야에서도 센서 기술과 컴퓨팅 기술의 발달 – 예를 들어 바이오인포메틱스의 DNA Array 칩이나 NASA 의 EOS (Earth Observing System) - 에 힘입어 데이터 수집 및 저장 능력이 크게 향상 되어서, 생성되는 데이터의 양은 가히 기하급수적이라 할 수 있다. 이러한 데이터의 증가 속도는 이미 현재의 단일 컴퓨터가 처리할 수 있는 속도를 능가했으며, 이로 인해 많은 귀중한 데이터가 분석되지 못하거나 심지어 검색되지도 못한 채 사장되고 있는 실정이다.

이러한 문제를 해결할 수 있는 방법은 고성능 다중 컴퓨터 시스템을 효율적으로 이용할 수 있는 병렬 처리 기법을 데이터 마이닝 기술에 접목 시키는, 병렬 데이터 마이닝 알고리즘을 개발하는 것이다. 병렬 데이터 마이닝 알고리즘은 실행 시키고자 하는 시스템의 특성을 제대로 고려해야만 좋은 성능을 얻을 수 있다. 즉, 어떠한 플랫폼을 타겟으로 병렬 데이터 마이닝 알고리즘을 개발할 것인가 하는 문제와 연결된다. 최근 PC 가격의 하락과 빠른 성능 향상으로 클러스터 시스템은 경제적으로 고성능의 계산 능력을 낼 수 있다. 특히, 데이터 마이닝과 같은 데이터 집약적인 응용 프로그램들이 주로 이용하는 정수 연산의 경우 PC 가 웍스테이션보다 더 좋은 성능을 보여 주고 있어 점점 더 두각을 나타내고 있다.

최근 데이터 양의 기하 급수적인 증가의 가속화로 인해 기존의 시스템보다 더 강한 연산 능력의 시스템이 요구되고 있다. PC 클러스터 시스템은 확장이 용이하기 때문에 단지 새로운 노드를 추가하는 것으로 이 문제에 효과적으로 대처할 수 있다. 그러나, 새로운 노드의 추가 시기는 기존의 시스템을 구축했을 때로부터 적게는 1 년 많게는 수년이 지나서 일 것이다. 다시 말해서, 새로운 노드를 추가할 시점에서는 기존의 노드들과 동일한 구성 성분들은 이미 단종되어 시스템의 이종화가 불가피하게 된다 [5].

이러한 이종성으로 인해 기존의 동종의 노드로 구성된 시스템에 기반해서 개발된 많은 병렬 데이터 마이닝 알고리즘을 이종의 노드로 구성된 시스템에 적용 하였을 경우 새롭게 추가된 더 나은 성능을 가지

는 노드들의 연산 자원의 낭비를 초래한다. 따라서, 이종의 노드로 구성된 시스템의 성능을 효율적으로 활용하기 위해서는 동종 시스템을 위해 개발된 병렬 알고리즘에 동적 부하 분산 알고리즘이 추가 되어야 함을 의미한다. 하지만 이러한 문제를 다룬 연구는 매우 적은 형편이며 그 결과도 아직 초기 단계에 있다 [5].

따라서 본 논문에서는 이종의 PC 클러스터 시스템에서 병렬 데이터 마이닝 알고리즘의 대표격인 병렬 연관 규칙 마이닝 알고리즘을 위한 동적 부하 분산 알고리즘을 개발한다.

병렬 연관 규칙 마이닝 알고리즘으로는 동등 클래스 (Equivalence Class, 줄여서 *eqclass*)에 기반을 둔 *Par-Eclat* 을 채택하였다 [7]. Par-Eclat 은 알려진 바와 같이 기존의 Apriori [1] 기반 병렬 알고리즘들 [2], [3], [4], [5]과 달리 반복적인 순환이 필요 없을 뿐만 아니라 단지 두 번의 DB scan 만이 요구된다. 그리고, 다빈도 항목집합 (Frequent Itemset)에 대한 전체 탐색 공간의 독립적인 처리가 가능하며 메인 메모리 내에 탑재 가능한 eqclass 단위로 클러스터의 각 노드에 분할함으로써 집합 메모리를 효율적으로 이용하며, 해쉬 트리 (hash tree)와 같은 복잡한 데이터 구조체를 탐색할 필요 없이 간단한 교집합 연산만이 요구되는 장점이 있다.

제안한 부하 분산 알고리즘은 이종의 클러스터의 모든 노드가 같은 시간에 다빈도 항목집합의 탐색을 끝낼 수 있도록 각 노드의 성능에 따라 작업량을 분배함으로써 시스템의 자원을 낭비 없이 효율적으로 이용할 수 있도록 한다. 이 때 작업량의 분배는 독립적인 처리의 장점을 이용하기 위해 eqclass 단위로 작업을 분배한다. 부하 분산 알고리즘으로써 각 노드의 성능을 측정하는 시점에 따라 선측정 방법과 실행중 측정 방법 모두를 제시하고 구현한다. 두 방법 중에서 측정 비용 차이로 인해 실행중 측정 방법이 선측정 방법보다 더 좋은 결과를 보여준다.

본 논문에서 제시한 성능 분석 방법은 주어진 이종 시스템 환경에서 제안된 부하 분산 알고리즘을 적용했을 때의 예상 수행 시간을 부하 분산이 없을 때의 수행 시간으로 나타냄으로써 어느 정도의 성능 향상을 기대할 수 있는지를 예측할 수 있다. 또한 예상 수행 시간과 실제 수행 시간의 상대 비교를 통해 제안된 부하 분산 알고리즘의 효율성을 측정할 수 있다.

논문의 구성은 다음과 같다. 우선 2 장에서는 eqclass 기반의 병렬 연관 규칙 마이닝에 대해 기술하고, 3 장과 4 장에서는 제안한 동적 부하 분산 알고리즘과 성능 분석법에 대해 설명한다. 그리고, 5 장과 6 장에서 모의 실험 결과와 결론을 제시한다.

2. 병렬 연관 규칙 마이닝

병렬 연관 규칙 마이닝 알고리즘으로 채택된 Par-Eclat 알고리즘은 다음과 같다. 자세한 내용은 [7]을 참고하길 바란다.

1) 전처리 단계에서 전체 F_2 공간을 하나의 prefix 를 공유하는 eqclass 로 분할 한다.
2) 각 eqclass 의 작업량 (work-load)을 계산한다. 작업량은 결합의 총수와 각 결합의 교집합 연산의 양에 비례하는데, 예상되는 결합의 총수를 eqclass 에 의해 생성되는 쌍의 총 개수, 즉, ${}_nC_2$ 로 근사화 시켰고 각 결합의 교집합 연산의 양은 eqclass 의 각 요소의 tid-list 의 평균크기로 근사화 할 수 있다. 다시 말해, 작업량은
$$\mathrm{wk}_i = \binom{num\_el_i}{2} \cdot tid_{avg} \quad (1)$$
로 계산 할 수 있다. 여기에서 wk_i 는 eqclass i 의 작업량을, num_el_i 는 eqclass i 의 멤버 요소의 수를, tid_{avg} 는 eqclass i 의 멤버 요소들의 tid-list 들의 평균 크기를 각각 나타낸다.
3) 위에서 계산된 작업량이 각 노드에게 골고루 분산 되어 전송 비용을 줄일 수 있도록 eqclass 를 각 노드들에게 분할한다.
4) 각 노드에 할당된 eqclass 에 포함된 아이템들 중에 자신의 로컬 DB 에 포함되지 않은 아이템들에 대한 tid-list 를 다른 노드의 DB 로부터 선택적으로 자신의 노드로 복사한다.
5) 각 노드에서 자신들에게 할당된 eqclass 에 대해 독립적으로, 각 eqclass 의 모든 해당 다빈도 항목집합을 찾는다 [6].

3. 동적 부하 분산 알고리즘

3.1. 작업 분배

PC 클러스터 시스템의 이종화에 따른 자원 낭비의 문제를 해결하기 위해서는 각 노드의 성능에 따라 작업량을 조절할 수 있는 동적 부하 분산 알고리즘을 설계해야 한다. 그 첫 번째 단계가 각 노드의 작업 처리 능력을 나타낼 수 있는 단위를 결정하는 것이다. 동적 부하 분산의 목적은 성능이 다른 모든 노드들이 같은 시간 안에 작업을 끝낼 수 있도록 작업량을 분배하는 것이다. 그러므로, 각 노드의 성능은 작업량과 시간의 관계를 나타내는 *throughput* = *work-load* / *runtime* 로 정의할 수 있다. 여기에서 throughtput, work-load, runtime 을 각각, v, l, t 로 나타내면 $v = l / t$ 로 나타낼 수 있다.

위의 식을 이용하면, 노드 j, $1 \le j \le n$ 의 작업량, l_j 는 $l_j = v_j t_j$ 로 정의할 수 있으며, v_j 는 노드 j 의 throughput 을, t_j 는 l_j 를 처리하기 위해 걸리는 시간을 의미한다. 그리고 전체 작업량, l_{tot} 는
$$\begin{aligned} l_{tot} &= l_1 + l_2 + \cdots + l_n \\ &= v_1 t_1 + v_2 t_2 + \cdots + v_n t_n \end{aligned} \quad (2)$$
로 나타낼 수 있다. 여기에서 각 노드의 독립적인 처리를 위해 작업 분배는 eqclass 단위로 이루어진다. 부하 분산 알고리즘의 목표는 노드가 동일한 시간 내에 자신에게 할당된 작업량을 끝내기를 원하는 것이므로,
$$t_1 = t_2 = \cdots = t_n = t^* \quad (3)$$
와 같이 설정 할 수 있으며, 위의 식을 이용하면 예상 수행 시간, t^*를

$$t^* = \frac{l_{tot}}{v_{tot}}, where\ v_{tot} = v_1 + v_2 + \cdots + v_n \quad (4)$$

로 찾을 수 있다.

마지막으로 각 노드에 대한 작업의 할당량, l_j 는 앞에서 구한 t^*와 측정된 v_j 를 이용하여 다음과 같이 계산할 수 있다.

$$l_j = v_j t_j = v_j t^* \quad (5)$$

3.2. 성능 측정

각 노드의 작업 할당량, l_j 를 구하기 위해서는 먼저 각 노드의 성능을 측정해야 한다. 이를 위해 *선측정 방법*과 *실행중 측정 방법* 두 가지 방법을 고려하였다. 선측정 방법은 자치적인 Eclat 단계 이전에 성능을 측정하는 방법으로써 우선 모든 노드에게 동일한 작업량을 할당하여 그 작업량이 끝나는 시간을 측정함으로써 각 노드의 throughput 을 계산할 수 있다. 이 방법은 간단한 반면, 가장 성능이 낮은 노드가 끝날 때까지 모든 노드가 기다려야 할 뿐만 아니라, 선측정을 위해 할당된 작업량이 클수록 선측정을 위한 비용이 증가하는 등의 문제점이 있다.

실행중 측정 방법은 자치적인 Eclat 의 수행 도중에 성능을 측정하는 방법으로써 초기에 전체 작업량 중 일부를 할당한 후 자치적인 Eclat 의 수행을 시작한지 일정 시점이 지난 후에 성능을 측정하여, 그 성능에 따라 남은 작업량을 분배하는 방법이다. 이 방법은 선측정의 비용을 제거할 수 있으며 단지 성능 정보의 교환 비용밖에 요구되지 않는다.

4. 성능 분석

동종의 클러스터 시스템을 위해 개발된 알고리즘을 이종의 클러스터 시스템에서 실행시키게 되면, 시스템의 전체 성능은 가장 낮은 성능을 가진 알고리즘에 의해 결정된다. 다시 말해, 전체 프로그램의 수행 시간은 가장 낮은 성능을 가진 노드의 수행 시간으로써 $t_{ov} = t_{max}$ 이다. 여기에서 t_{ov} 는 전체 시스템의 총 수행 시간이고 t_{max} 는 가장 낮은 성능을 가진 노드의 총 수행 시간이다. 빠른 노드들과 느린 노드들 사이의 성능 차이가 크면 클수록, 빠른 노드들의 자원이 낭비되기 때문에 시스템의 효율성은 더욱 떨어지게 된다.

이런 문제점을 해결하기위해 제안된 동적 부하 분산 알고리즘의 성능을 측정하기 위해 우선 각 노드의 성능, 즉 throughput (v_j)을 가장 낮은 노드의 throughput (v_{min})과 비교하여 다음과 같이 나타낼 수 있다.

$$v_j = r_j \cdot v_{min} \quad (6)$$

여기에서, r_j 는 v_j 와 v_{min} 사이의 비율이다. 부하 분산 알고리즘에 의해 결정된 예상 수행 시간 t^*는 수식 (4)와 (6)에 의해

$$t^* = \frac{l_{tot}}{\sum v_j} = \frac{l_{tot}}{\sum r_j \cdot v_{\min}} \quad (7)$$

또한 $t_{max} = l_j / v_{min}$ 이고, 부하 분산 알고리즘을 적용하지 않을 경우 $l_j = l_{tot} / n$ 이므로

$$t^* = \frac{n}{\sum r_j} \cdot t_{\max} \quad (8)$$

가 된다. 수식 (8)에서 $1 \le j \le n$ 에 대해, $r_j \ge 1$ 이기 때문에,

$$t^* \le t_{\max} \quad (9)$$

이다. 예를 들어 4 대의 노드로 구성된 클러스터 시스템에 성능이 4 배 향상된 새로운 노드를 4 대 더 추가했을 경우 $t^* = 8 / (4*4 + 4*1) \cdot t_{max} = 0.4\ t_{max}$ 로써 부하 분산 알고리즘을 적용하지 않았을 때에 비해 이론적으로 40%의 수행 시간밖에 걸리지 않는다.

t^* 는 부하 분산 알고리즘으로 인해 발생하는 비용을 고려하지 않은 것이다. 하지만, 선측정을 위해 할당된 작업량이 전체 작업량에 비해 충분히 작을 경우 추가되는 비용을 무시할 수 있다. 또한 실행 중 측정 방법의 경우 실행 중 일정 시점에서 각 노드의 throughput 정보의 교환 비용밖에 요구 되지 않으므로 그 비용을 무시할 수 있다.

5. 결과 및 고찰

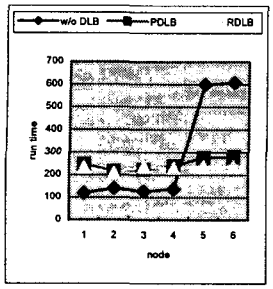
모든 실험은 Fast Ethernet 100Mbps 네트워크로 연결된 4 대의 Pentium III 850MHz 노드 (노드 1-4)와 2 대의 Pentium II 333MHz 노드 (노드 5-6)로 구성된 이종의 PC Cluster system 에서 이루어졌다.

실험에서 사용된 데이터베이스는 [1]에서 소개된 방법으로 생성한 3 종류의 서로 다른 합성 데이터 베이스를 이용하였다. 표 1 에서 T 는 트랜잭션의 평균 크기를, I 는 최대 잠재 다빈도 항목집합의 평균 크기를, D 는 트랜잭션의 수를 나타낸다. 최대 잠재 다빈도 항목집합의 수를 나타내는 L 은 2000 으로, 아이템의 수를 나타내는 N 은 1000 으로 설정하였다. 표 1 의 마지막 열에는 실험에서 각 데이터베이스에 사용된 support 를 나타내었다.

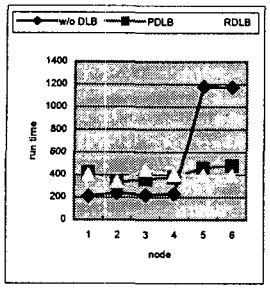
표 1. 합성 데이터베이스

Name	T	I	Size	Support
T10I4D2048K	10	4	107.3MB	0.025%
T15I4D2048K	15	4	147.1MB	0.5%
T20I6D2048K	20	6	187.6MB	0.1%

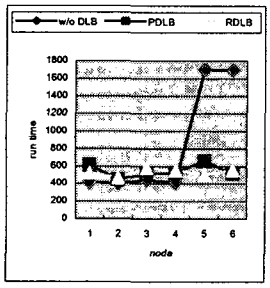
그림 1 의 (a), (b), (c)는 각각의 합성 데이터베이스들에 대한 각 노드의 수행시간을 측정한 것이다. 이 때 각 DB 에 대해 표 1 에 있는 support 를 적용하여 실험한 것이다. 그림에서 두 가지 주시 해야 할 것이 있는데, 첫번째는 각각의 방법에서 각 노드의 실행 시간에 대한 분포이다. 부하 분산 알고리즘을 적용하지 않았을 경우 (w/o DLB), 빠른 노드에 해당하는 노드 1-4 와 느린 노드에 해당하는 노드 5-6 의 수행시간의 차이로 인한 비대칭이 큰 분포를 보이고 있다. 이러한 비대칭은 구성이 다른 노드들의 성능 차이가 클수록 더 커지며, 시스템의 효율성을 더욱 악화시킨다. 하지만 부하 분산 알고리즘을 적용하였을 경우 (PDLB 와 RDLB) 이러한 비대칭이 제거 되어 거의 동일한 수행시간을 보여줌을 알 수 있다. 두 번째로 주시해야 할 사항은 각각의 방법에서 각 노드에 대한 실행 시간 중 최대값이다. 이 값들이 바로 알고리즘의 전체 수행시간을 의미한다. 그림 1 의 (d)는 이 값들을 따로 분리해서 나타낸 것이다. 그림에서 볼 수 있듯이



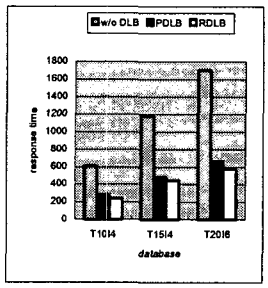
(a) T10I4D2048K



(b) T15I4D2048K



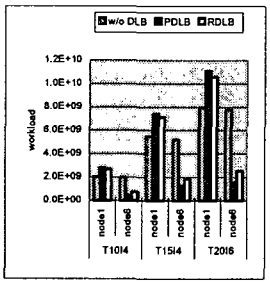
(c) T20I6D2048K



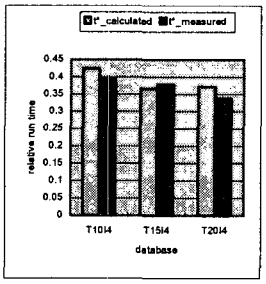
(d) Response times

그림 1. 성능 비교

w/o DLB 가 가장 느린 결과를 보여주고 있으며, 다음은 PDLB 이고, RDLB 가 가장 빠른 전체 수행 시간을 보여준다. RDLB 가 PDLB 보다 빠른 이유는 PDLB 의 경우 선측정 비용이 드는데 반해, RDLB 는 실행 중단 한번 성능 정보 (throughput)를 측정하고 교환하는 비용밖에 요구되지 않기 때문이다.



(a)



(b)

그림 2. (a) 작업량 분포 (b) 상대 수행 시간

그림 2 (a)는 빠른 노드와 느린 노드를 각각 대표하는 노드 1 과 노드 6 의 작업량의 분포를 보여주고 있다. 부하 분산 알고리즘을 적용하지 않았을 경우 노드 1 과 노드 6 에 노드들의 성능과 상관 없이 거의 동일한 작업량이 분배 되었지만, 부하 분산 알고리즘을 적용하였을 경우, 각 노드의 성능에 따라 빠른 노드인 노드 1 에 더 많은 작업량이 분배 되었다.

마지막으로 그림 2 (b)는 수식 (8)에 의해 계산된 부하 분산 알고리즘이 없을 때의 수행 시간에 대한 부하 분산 알고리즘 (RDLB)이 있을 때의 상대 예상 수행 시간과 실제 측정된 상대 수행 시간을 나타내고 있다. 이 값들은 t_{max}, 즉, 가장 느린 노드의 수행시간에 대한 상대적인 수행 시간이다. 이 결과는 제안한 알고리즘의 정확성을 암시한다.

6. 결론

본 논문에서는 이종의 PC 클러스터 시스템에서 동작하는 병렬 연관 규칙 마이닝을 위한 부하분산 알고리즘을 제안하였으며 그 실용성을 평가 하였다. 부하 분산 알고리즘에 사용될 각 노드들의 성능은 작업의 가장 기본 단위인 시간당 수행할 수 있는 결합 (교집합 연산)의 양으로 정의하였으며, 각 노드에 대한 작업량은 eqclass 단위로 분배하였다. 부하 분산을 위해 구현된 두 가지 방법 중에서 선측정 방법은 정확한 성능 측정이 가능하지만 실제 작업을 분배하기 전에 성능을 측정해야 하는 비용으로 인해 실행 중 측정 방법보다 더 많은 총 수행 시간이 요구되었다. 실험 결과에서 실제 얻어진 수행 시간이 성능 분석을 통해 얻어진 예상 수행 시간에 부합함을 알 수 있었으며, 각 노드의 성능에 따른 작업량의 적정한 분배를 통해 전체 시스템이 효율적으로 이용되고 있음을 확인하였다. 제안된 부하 분산 알고리즘은 작업량의 정의를 적절히 수정함으로써 연관 규칙 마이닝 이외의 다른 마이닝 기법에서도 활용될 수 있을 것으로 기대한다.

감사의 글

본 연구는 교육부 두뇌한국 21 (BK21) 정보기술사업단의 지원에 의한 것입니다.

참고문헌

[1] R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules," Proc. the 20th Int'l Conf. VLDB (1994) 487–499

[2] R. Agrawal and J.C. Shafer. "Parallel Mining of Association Rules," IEEE Trans. Knowl. and Data Eng., Vol. 8, No. 6 (1996) 962–969

[3] D. Cheung, Y. Xiao. "Effect of Data Skewness in Parallel Mining of Association Rules," Proc. Pacific-Asia Conf. KDD. Lecture Notes in Computer Science, Vol. 1394, Springer-Verlag, Berlin Heidelberg New York (1998) 48–60

[4] E.H. Han, G. Karypis, and V. Kumar. "Scalable Parallel Data Mining for Association Rules," Proc. ACM SIGMOD Conf. Management of Data (1997) 277–288

[5] M. Tamura and M. Kitsuregawa. "Dynamic Load Balancing for Parallel Association Rule Mining on Heterogeneous PC Cluster System," Proc. the 25th VLDB Conf. (1999) 162–173

[6] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W.Li. "New Algorithms for Fast Discovery of Association Rules," Proc. the 3rd Int'l Conf. KDD (1997) 283–286

[7] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. "Parallel Algorithms for Discovery of Association Rules. Data Mining and Knowl. Discovery," An Int'l J., Vol. 1, No. 4 (1997) 343–373