분산 데이터 베이스의 효율적 설계를 위한 모델과 알고리즘

서울대학교 경영학과 노 상규

Abstract

Distributed database systems can yield significant cost and performance advantages over centralized systems for geographically distributed organizations. The efficiency of a distributed database depends primarily on the data allocation (data replication and placement) and the operating strategies (where and how retrieval and update query processing operations are performed). We develop a comprehensive distributed database design model that treats data allocation and operating strategies in an integrated manner, explicitly modeling their interdependencies for both retrieval and update processing. Our model includes data replication, a concurrency control mechanism, data reduction by semijoin, join node selection, and join ordering. We include two evaluation criteria, minimization of total operating cost and minimization of average response time. To address tractability, we develop a nested genetic algorithm to solve this problem formulation.

1. 서론

지역적으로 분산된 조직들은 지역내 작업과 조직 전반에 걸친 정보의 공유를 효율적으로 지원해야만 한다. 상업적인 분산 데이터 베이스 관리 시스템(distributed database management system)의 등장과 함께[Ricciuti, 1993, Richter, 1994, The, 1994], 분산 데이터 베이스 시스템(distributed database system)은 일반화 되었다. 분산 데이터 베이스 시스템은 사용자에게 다른 지역에 있는 기업 데이터 베이스에 대한 액세스를 제공해 준다. 그러한 시스템들은 집중화된(centralized) 데이터 베이스 시스템에 비해 지역적으로 분산된 기업에게 비용과 성과 면에서 상당한 이점을 준다. 이러한 이점으로는 시스템 성과의 개선, 시스템 비용의 감소, 그리고 데이터 사용 가능성의 개선을 포함한다[Ozsu and Valduriez, 1991a, 1991b].

특정 데이터 전송 속도와 용량을 갖는 링크(link)에 의해 연결된 노드(node, 처리와 저장 능력을 갖춘 컴퓨터)로 구성되어지는 하나의 컴퓨터 네트워크가 주어질 때, 데이터와 처리 능력의 신중한 배치에 의해 매우 효율적이고 응답이 빠른 시스템을 만들 수가 있다. 그러나, 데이터의 부적절한 복사나 배치 또는 처리 능력의 부적절한 이용은 많은 비용과 좋지 않은 시스템 성과를 유발시킬 수가 있다[Ozsu and Valduriez, 1991b]. 분산 데이터 베이스 설계distributed database design)는 데이터 배분(data allocation)과 운영 전략(operating strategies)의 두 가지 면이 있다. 데이터 배분은 배분할 데이터 단위(파일 프래그먼트(file fragment)라고 불린다.)의 결정과 네트워크내의 노드에 각 데이터 단위의 배치를 포함한다. 검색 효율을 향상시키기 위해서 동일한 데이터가 여러 개의 노드에 중복적으로 배분되어 질 수 있다(즉, 데이터는 복사되어 질 수 있다.). 물론, 그러한 중복성은 갱신 비용을 증가시킨다.

운영 전략은 작업 배분(operation allocation; 또는 질의 최적화(query optimization))과 동시성 통제 전략(concurrency control strategy)을 포함한다. 작업 배분은 검색과 처리 작업들이 이루어지는 장소와 방법, 그리고 시기를 정의한다[Yu and Chang.1984]. 검색 작업들은 필요한 데이터를 보유하고 있는 노드에서 이루어 져야만 한다. 처리 작업들은 어떠한 노드에서도 이루어 질 수가 있지만 만약에 데이터가 그 처리 노드에 들어 있지 않다면 필요한 데이터들은 통신 네트워크를 통해서 그 노드로 보내져야 한다. 작업들이 이루어지는 순서는 성과에 중요한 영향을 미칠 수 있다. 필요한 처리의 양을 줄이기 위해 선택(select)과 프로젝트(project)작업은 언제나 조인(join) 작업 이전에 이루어진다. 그러나, 조인 작업들이 이루어지는 순서와 데이터 감소 전략의 사용은 결정되어야 한다.

동시성 통제 체제는 특별히 여러 개의 데이터 복사본이 존재할 때, 갱신 작업이 정확하고 일관성 있게 이루어지는 것을 보장하는 책임을 진다[Bernstein and Goodman, 1981]. 갱신 작업은 실제로 영향을 받는 데이터의 복사본을 보유하고 있는 모든 노드에서 이루어 져야만 한다.

데이터 배분과 작업 배분은 상호 의존적인 문제들이다[Apers, 1988]. 파일 프래그먼트들의 최적 집합과 그들의 최적 배분은 검색이 어떻게 처리되는지에 의존한다(i.e., 작업 배분). 그러나, 최적 작업 배분은 파일 프래그먼트들이 어디에 위치하고 있는지에 의존한다(i.e., 데이터 배분). 이 분야의 초기 연구는 데이터 중복이 없거나 각 검색에 사용될 데이터 복사본이 사전에 선택될 것이라는 가정하에 데이터 배분에 초점을 두거나 데이터 배분이 되어 있다는 가정 하에 작업 배분(검색 최적화)에 초점을 두었다. 이 연구의 대부분은 갱신과 동시성 통제 체제의 효과를 무시했다.

본 연구에서는 하나의 모델에서 데이터 배분과 작업 배분을 결합한다. 이 모델은 데이터의 복사, 갱신 작업, 동시성 통제 체제, 데이터의 감축, 조인 노드 선택, 그리고 조인 순서 결정을 포함하며 운영비용(operating cost)과 응답 시간(response time)을 모두 평가한다.
이러한 모델을 해결하기 위해 GA(genetic algorithm)에 기초한 해법을 개발하고 이를 예제에 적용한다. 그 알고리즘은 최소 운영비용이나 최소 응답 시간 기준에 의거해서 효율적인 데이터와 작업 배분을 선택한다.

논문의 나머지 부분들은 다음과 같이 구성되어 있다. 다음 장은 데이터 배분과 작업 배분에 초점을 맞추어 분산 데이터 베이스 시스템의 기본적인 배경을 제시하고 그 다음 장은 이전의 연구를 간단하게 재검토한다. 그리고 나서 본 연구의 모델과 알고리즘을 제시한다. 마지막으로 앞으로의 연구 방향에 대해 간단히 언급한다.

2. 분산 데이터 베이스 시스템

분산 데이터 베이스 시스템에서는 하나의 개념적 데이터베이스를 이루는 데이터가 컴퓨터 네트워크의 여러 노드에 유지되어진다. 노드에 데이터를 배분하는 과정은 분산 설계(distribution design) 또는 데이터 배분으로 불려 진다[Ceri et al., 1987, Ozsu and Valduriez, 1991a]. 데이터 배분이 되면, 사용자 검색과 갱신이 처리되어야 한다. 질의(query)는 어느 한 노드에서 발생되고 어떤 노드에 저장된 데이터를 갱신하거나 검색한다. 검색이 처리되는 방법과 시간, 장소를 결정하는 과정은 질의 최적화 또는 작업 배분으로 불려 진다. 동시성 통제 체제는 갱신 처리의 제약들을 명시한다.

전형적으로 데이터 배분과 동시성 통제 전략은 설계 시에 결정되고 드물게 변

한다(데이터 이동 전략에 대한 연구[Gavish and Sheng, 1990]가 있지만 분산 시스템 운영의 이러한 면은 이 논문의 영역 밖이다.). 작업 배분은 일반적으로 컴파일(compile) 시점(e.g., R*[Lohman et al., 1985]이나 실행(run) 시점(e.g., Distributed INGRES[Epstein et al., 1978])에서 분산 데이터 베이스 관리 시스템 내의 질의 최적화 기능에 의해서 이루어진다. 우리는 설계 시에 각 질의의 효율적인 작업 배분을 결정하는 것이 중요하다는 것을 논의한다. 이것은 설계자들이 시스템 부하를 평가하고, 질의 처리 전략(query processing strategy)들을 미리 컴파일할 수 있게 해 준다. 포괄적으로 최적화된 질의 처리 전략들은 실제로 하나 하나씩 최적화된 질의 처리 전략보다 효율적일 수 있다.

데이터 배분과 작업 배분 결정이 평가되어지는 기준은 주로 시스템 비용과 응답 시간에 관계되어 진다. 시스템 비용은 데이터 저장, 네트워크 통신, 그리고 노드 내 처리를 포함한다. 변동비로서 이러한 비용의 결정은 그 자체로서 하드웨어의 이용도, 운영의 실제 변동비(전기, 인건비 등), 그리고 투자의 회복(초기 비용, 이자, 감가 상각 등)과 같은 요소들을 고려해야 하는 어려운 문제이다.

응답 시간은 데이터 베이스 질의가 시스템에서 소비하는 기대 시간이다. 그것은 노드 내 데이터 처리와 데이터 전송에서 일어나는 처리 시간과 지연들을 포함한다. 응답 시간은 일반적으로 포아송 분포 도착 과정과 지수 분포 서비스 시간을 가정하는 개방 대기 행렬 네트워크를 사용해서 평가된다[Kleinrock, 1975, Cornell and Yu, 1989].

설명에 도움을 주기 위해서 본부와 세 개의 지역 사무실을 가지고 있는 은행을 생각해 보자. 더 나아가서 각 지역은 하나의 컴퓨터 시스템을 가지고 완전하게 연결된 네트워크를 이룬다고 가정하자. 각 컴퓨터는 중앙 처리 장치와 디스크 용량과 단위 비용에 의해 묘사된다. 네트워크에서 각 링크는 속도, 용량, 그

리고 단위 전송 비용에 의해 묘사된다. 데이터 베이스는 Customer, Account, 그리고 Transaction의 3 테이블로 구성된다고 가정하자(그림1). 각 고객은 거래(예금과 인출)를 하는 하나 이상의 구좌를 가지고 있다. 각 고객은 그들의 은행 업무의 대부분을 수행할 지역 사무실(즉, 구좌들이 개설되어 있는 사무실)을 가지고 있다. 물론, 고객은 어느 지역 사무실에나 다 갈 수 있다. 각 지역 사무실은 어떤 지역 고객에 대해서도 거래들을 처리해 줄 수 있어야만 한다. 더 나아가서 지역 사무실과 본부는 여러 고객과 구좌, 그리고 거래에 대한 데이터에의 액세스가 필요하다.

그림2는 검색과 갱신 질의의 예를 보여준다. 각각은 선택 기준과 빈도에 따라 각 지역에서 실행된다. 예를 들어서, 질의 R1 은 지역1 구좌들(즉, br-i='지역1')을 선택하면서 하루에 한 번 본부에서 실행되어질 수 있다. 또한 지역2 구좌들을 선택하면서 지역2에서 한 달에 한 번씩 실행될 수 있다. 분산 데이터 베이스 시스템에서는 알려진 질의들의 효율적인 수행을 위해서 데이터와 작업들을 배분해야만 한다.

2.1.데이터 배분

데이터 배분은 분산 데이터 베이스 시스템의 각 노드의 데이터베이스 구조(subschema)를 생성한다[Ceri et al.,1987]. 데이터 배분에 앞서서, 배분할 데이터의 단위들이 결정되어야 한다. 이 과정은 프래그멘테이션(fragmentation)이라고 불려 진다[Navathe et al., 1984]. 프래그멘테이션은 수직적인 것과 수평적인 것의 두 가지 형태가 있다. 수평적인 프래그멘테이션은 한 선택 조건을 만족시키는 한 파일의 레코드들을 그룹화 한다[Ozsu and Valduriez, 1991b]. 수직적인 프래그멘테이션은 함께 액세스될 가능성이 큰 파일의 속성들을 그룹화 한다[March, 1983, Navathe et al., 1984].

그리고 나서, 프래그먼트는 노드에 배분되어진다[Dowdy and Foster, 1982]. 프래그먼트 배분은 중복되게 또는 중복되지 않게 이루어 질 수 있다.

검색 요구에 대한 운영비용과 응답 시간은 중복적 배분에 의해 줄어들 수가 있다. 각 프래그먼트의 복사본을 그것을 참조하는 각 노드에 중복적으로 배분하는 것은 모든 검색 질의의 노드 내 처리를 가능케 한다. 그러나, 그러한 데이터의 복사는 모든 참조되어지는 프래그먼트의 복사본이 갱신되어 져야만 하기 때문에 갱신 질의의 운영비용과 응답 시간을 증가시킨다. 갱신 비용에 대한 중복적 배분의 정확한 효과는 분산 데이터 베이스 운영 전략의 한 요소인 동시성 통제 체제에 의해 결정된다.

[Ram and Marsten, 1991, Ram and Narasimhan, 1990, 1994].

2.2.운영 전략

위에서 논의된 바와 같이, 운영 전략은 작업 배분과 동시성 통제 체제의 두 가지 요소로 구성된다.

작업 배분은 복사본 선택(copy identification), 감축(reduction), 그리고 조립(assembly)의 3단계를 포함한다[Yu and Chang, 1984].

복사본 선택(또는 유형화(materilization))에서 질의에 의해 참조되어지는 각 프래그먼트의 하나 이상의 복사본들이 처리를 위해 선택되어진다. 적절한 프래그먼트 복사본의 선택은 전체적인 질의 처리 비용을 결정하는데 중요한 역할을 할 수 있다[Martin et al., 1990, Yu and Chang, 1983, 1984].

감축은 결합될 프래그먼트들이 서로 다른 노드에 저장되어 있는 조인 질의

(join query)에만 적용된다. 감축 단계에서 조인을 하기 위해 전송되어야만 하는 데이터의 양을 줄이는데 세미조인(semijoin)이 사용된다. 감축자(reducer)-->피감축자(reducee)로 표시되는 세미조인에서는 감축자의 유일한 조인 속성치(attribute value)가 피감축 프래그먼트를 보유하고 있는 노드로 전송된다. 피감축자의 레코드는 그것의 조인 속성이 전송된 조인 속성치들 중에 하나와 일치하면 선택되어진다. 단지 선택된 레코드들만이 감축자를 보유하는 노드로 전송되어지고 조인은 거기서 수행되어진다.

만약 피감축자에 있는 모든 레코드들이 선택되어 진다면, 어떠한 데이터 감축도 이루어지지 않고 전체 프래그먼트가 감축자의 노드에 보내어 진다. 이러한 경우에, 세미조인 전략은 유익하지(beneficial) 못할 것이다. 다시 말해서, 그것은 조인을 수행하기 위해서 전송될 데이터의 양을 줄이지 못할 것이다. 분산 질의 최적화(distributed query optimization)에서의 대부분의 연구는 세미조인이 유익한 상황을 찾는데 집중되었었다.[예를 들면, Apers et al., 1983, Bernstein and Dhiu, 1981, Hevner and Yao, 1979].

조립에서 데이터는 결과 노드에 보내어 지고(만약에 그들이 아직도 거기에 있지 않다면) 마지막 처리(예를 들면, 정렬과 총계)가 수행되어진다. 이전의 연구는 일반적으로 세미조인에 의해 감축된 파일이 결과 노드로 전송되어져서 모든 조인이 거기서 수행되어 진다고 가정한다. 더 나아가서, 이전의 연구는 노드 내 처리 비용을 무시하여 결과적으로 조인 순서에 대한 고려를 불필요하게 했다. 그러나, 조인이 수행되는 노드와 조인 순서는 전체적인 질의 처리 비용과 응답 시간에 중대한 영향을 미친다[Mishra and eich, 1992]. 이 연구는 조인노드 선정과 조인 순서 결정을 데이터와 작업 배분의 포괄적인 모델에 통합시킨다.

동시성 통제 체제는 갱신 작업이 이루어지는 방법을 명시해 준다. 특히, 복사된 데이터가 일관성 있게 유지되는 것을 보장한다. 많은 분산 동시성 통제 체제

들이 제안되어져 왔다(예를 들면, 2단계 잠금(2 phase locking)[Mohan et al., 1986], 타임 스탬프(timestamp)[Bernstein etal., 1980], 낙관적(optimistic)[Ceri and Owicki, 1981]). 이 중 2단계 잠금(2PL)이 가장 일반적으로 사용되어진다. 2PL의 한가지 예인 분산 2PL이 이 연구에서 모델화 되어 진다. 다음 장은 분산 데이터 베이스 설계와 분산된 질의 최적화에 대한 이전의 연구들을 간단하게 살펴 본다.

3. 이전의 연구

몇몇 연구자들이 통합된 데이터와 작업 배분 문제를 해결하기 위한 모델들을 개발해 왔다[Apers, 1988, Blankinship et al., 1991, Cornell and Yu, 1989]. 그러나 그들은 데이터 복사나 갱신 질의, 그리고 그에 따른 동시성 통제 체제의 효과를 고려하지는 않는다. 또 그들은 세미조인과 조인 순서의 효과도 고려하지 않는다. 파일이 배분의 부적절한 단위라는 것을 염두에 두면서 Apers[1988]는 배분을 위한 파일 프래그먼트들을 결정하는 접근법을 개발했다. Blankinship 등은 [1991] 데이터와 작업의 배분은 비용과 응답 시간의 두 가지 기준에 의해서 평가되어야 한다고 말한다.

Cornell과 Yu[1989]는 먼저 질의를 단계로 나누고 각 노드에 파일과 질의 단계들을 배분하는 모델을 개발했다. 그들의 비용 모델은 단지 통신비용만을 포함하기 때문에 단순하다. 그러나 그들의 제약식들은 노드 내 데이터 저장, 입출력, 그리고 처리 능력과 통신 능력을 포함하기 때문에 포괄적이다. 추가로 그들의 응답 시간 분석은 대기 행렬 지연을 포함한다. 그러나, 다시 말하지만 그들은 데이터의 복사나 갱신의 효과를 고려하지 않는다.

Ram과 Narasimhan[1990]은 데이터 복사와 동시성 통제 체제, 즉 하나의 공유

된 네트워크 디렉토리를 가지는 집중(centralized) 2PL을 포함하는 모델을 형성한다. 그들의 모델은 노드 내 메시지 처리 작업에서 대기 행렬 지연들을 포함한다. 그들은 파일들은 독립적으로 액세스되고 일단 액세스된 파일들은 저장 노드로부터 모든 처리가 이루어지는 질의 실행 노드로 보내어 진다고 가정한다. 그러나, 복잡한 분산 질의를 처리하는데 있어서 데이터를 필요로 하는 노드에 결과를 보내기 전에 중간 노드에서 처리를 하는 것이 더 효율적일 수 있다. 더 나아가서, 메시지를 제외하고는 노드 내 처리는 무시된다. Ram 과 Narasimhan[1994]은 주복사본(primary copy) 2PL 동시성 통제 체제에 기초한 비슷한 모델을 형성한다.

March 와 Rho[1995]는 데이터 복사, 갱신 질의 그리고 동시성 통제 체제를 포함하는 모델을 개발했다. 그들의 모델은 데이터와 작업 배분을 하나의 통합된 방식으로 다룬다. 그들의 비용 모델은 통신 비용들뿐만 아니라 노드 내 저장, 입출력, 그리고 중앙 처리 장치 처리 비용을 포함한다. 작업은 어떠한 노드에도 배분될 수 있으나, 그들은 모델에서 조인 순서는 미리 정해진다고 가정한다. 더 나아가서, 그들의 모델은 세미조인을 포함하지 않는다.

위에서 논의된 바와 같이, 조인 순서와 세미조인의 사용은 성과에 중요한 영향을 미칠 수 있다. 세미조인을 다루는 분산 질의 최적화에서의 연구는 [Apers et al., 1983, Bernstein and Chiu, 1981, Hevner and Yao, 1979, Yoo and Lafortune, 1989] 데이터 배분이 주어진다고 가정한다.

본 논문에서는 March와 Rho[1995]의 기본적인 접근법을 확장하여 세미조인에 의한 감축과 조인 순서의 결정을 포함하는 포괄적인 작업 배분 모델을 통합한다. 본 논문에서는 배분을 위한 파일 프래그먼트 결정에는 Apers[1988]의 접근법을 질의분해(query decomposition)에는 Cornell 과 Yu[1989]의 접근법을 채택

한다. 본 논문에서는 검색과 갱신 질의를 모두 포함하는 노드 내 데이터 처리에서와 통신에서의 비용과 대기 행렬 지연을 모델화 하며 총 운영비용의 최소화와 평균 응답 시간의 최소화라고 하는 두 가지 평가 기준을 포함한다. 다음 장에서는 모델과 그 해법을 제시한다.

4. 데이터와 작업 배분을 위한 모델과 알고리즘

본 연구에서는 성과를 최적화시킬 질의와 함께 통합 데이터 베이스 구조(global database schema)가 주어지는 것으로 가정한다. March와 Rho[1995] 그리고 Apers[1988]와 마찬가지로 먼저 질의들의 선택 기준으로부터 배분을 위한 min-term 프래그먼트들을 결정한다. 예를 들어서 그림 2의 검색 질의인 R1, R2, 그리고 R3에 입각해서 그림 1에 있는 각 데이터들은 지역에 따라 3개의 프래그먼트들로 수평적으로 나누어 질 수가 있다(예를 들면, Customer를 Customer1, Customer2, Customer3으로 나눈다.).

그리고 나서 질의를 min-term 프래그먼트에 따라 하부 질의(subquery)로 변경시키는데 이러한 하부 질의는 다음에 질의 단계(query step)로 분해되어진다. 예를 들어서, 지역 1로부터의 검색 질의 R3는 단순히 상응하는 min-term 프래그먼트(즉 Account1)를 검색하겠지만, 본부로부터의 R3는 세 개의 프래그먼트 모두의 결합을 요구할 것이다(즉, Account1, Account2, Account3). 그러므로, R3는 필요한 프래그먼트들에 입각한 3개의 하부 질의들(즉, R1.1, R1.2, R1.3)로 분해되어 진다. 질의 단계는 수행되어져야만 하는 실제 검색과 처리는 물론 전송될 필요가 있는 모든 메시지를 포함한다. 만약에 질의에 필요한 데이터가 데이터를 필요로 하는 노드에 저장되어 있지 않으면, 그 노드는 데이터가 검색될 노드에 메시지들을 보내야만 한다. 우리는 2PL 동시성 통제 체제를 가정하기 때문에, 갱신 질의는 일군의 메시지를 필요로 한다.

따라서 총 운영비용이나 평균 응답 시간을 최소화하기 위해서 필요한 일들은

(1) min-term 프래그먼트를 노드에 배분하고(중복적 데이터 배분)

(2) 노드에 검색 질의 단계를 배분하고(복사본 선택), 각 조인질의를 위해서

(3) 유익한 세미조인을 결정하고

(4) 조인 순서를 결정하고

(5) 노드에 조인 작업을 배분한다(조인 노드의 선택).

Cornell과 Yu[1989]와 March와 Rho[1995]와 마찬가지로 a(k,m)와 b(k,m)은 질의 k의 단계 m에 의해서 참조되어지는 파일 프래그먼트로서 정의되어진다. 메시지와 노드 내 선택(select)과 프로젝트(project) 단계에서는 단지 하나의 파일이 참조되어지며, 그러한 단계들에는 b(k,m)은 존재하지 않는다. 프래그먼트들을 결합하는 단계(조인과 유니언)에 있어서 a(k,m)과 b(k,m)은 이전의 선택과 프로젝트, 세미조인 또는 프래그먼트를 결합하는 단계에서 발생되어진 일시적인 파일들이다. 다시, Cornell과 Yu를 따라 L_i 는 파일 프래그먼트 i의 크기로 정의되고(문자 수로) L^M은 메시지의 크기로서 정의되어진다. 각 파일 프래그먼트의 크기는 문제를 기술하는 변수들로부터 계산된다. 각 일시적 파일의 크기는 그들을 생성하는 선택과 프로젝트, 세미조인, 조인 작업의 조건들로부터 예측되어진다(예를 들면, Gardy 와 Peuch[1989]). 더 나아가서 우리는 node(k)를 질의 k 의 실행노드(origination node)로서 정의하고 node(k,m)을 질의 k의 m 단계가 수행되는 노드로서 정의한다. 그리고 node(i)는 프래그먼트 i가 액세스되어지는 노드로서 정의한다. 조인 단계의 node(k,m)과 검색의 메시지 단계의 node(a(k,m))은 각각 조인 노드 선택과 복사본 선택을 나타낸다. 끝으로, copy(i,t)는 프래그먼트 배분을 나타내고 프래그먼트 i가 노드 t에 저장되면 그 값은 1이고 그렇지 않으면 0이다. 세미조인과 조인 순서는 각 질의가 어떻게 분해되는지를 결정하기 때문에 그들은 비용 모델에는 나타나지 않는다.

4.1.총 운영비용 모델

첫번째 성과 모델은 통신, 디스크 입출력, 중앙 처리 장치 처리, 그리고 저장을 포함하는 총 운영비용을 최소화하기 위하여 설계되어 진다.

$$Min\ Cost = \sum_{k} f(k) \sum_{m} (COM(k,m) + IO(k,m) + CPU(k,m)) + \sum_{t} STO(t)$$

여기서 f(k)는 단위 시간당 질의의 실행 빈도이고, COM(k,m), IO(k,m), 그리고 CPU (k ,m)은 각각 질의 k의 m 단계를 위한 통신, 디스크 입출력 그리고 중앙 처리 장치 작업의 비용이고 STO(t)는 단위 시간당 노드 t에서의 저장 비용이다. 이러한 비용 요소들에 대한 표현들은 부록 1에 요약되어 진다.

실행 가능성(feasibility)을 가지기 위해서 데이터와 작업의 배분은 시스템 자원 능력을 초과해서는 안된다. 본 모델에서는 통신 링크, 디스크 입출력, 중앙 처리 장치, 그리고 저장 능력을 제약 조건으로 고려한다.

4.2.평균 응답 시간 모델

질의 k의 평균 응답 시간은 3부분으로 분해되어질 수 있다: 통신 ($R_{com}(k)$), 디스크 입출력 ($R_{IO}(k)$), 그리고 중앙처리장치 ($R_{CPU}(k)$). 그리고 목적함수식은:

$$Min\ R_t = \frac{\sum_{k} f(k)(R_{COM}(k) + R_{IO}(k) + R_{CPU}(k))}{\sum_{k} f(k)}$$ 이다.

우리는 통신 링크와 디스크, 그리고 중앙 처리 장치의 모델을 위해서 M/M/1 대기행렬 모델을 가정했다. 위의 응답 시간 요소들의 표현은 부록 2에 요약된다.

5. GA(Genetic Algorithm) 해법 과정

이 영역에서 연구자들이 직면하는 어려움들 중에 하나는 해결 가능성(tractability)이다. 데이터와 작업 배분은 상호 의존성이 있는 문제들이고, 그들 각자는 NP-hard이다[Eswaran, 1974, Hevner, 1979]. 해결 가능성의 문제를 풀기 위해, 우리는 GA(Genetic Algorithm)에 기초한 해법 과정을 개발했다 [Goldberg, 1989, Davis, 1991]. GA는 몇 가지 이유때문에 선택되어 졌다. 첫째, GA는 분산 데이터 베이스의 설계를 포함하는 복잡하고, 순열 조합적인 실제 세계의 문제들에 성공적으로 적용되어졌다 [March and Rho, 1995]. 둘째, GA는 불연속적이고, 여러 모드를 가지며, 잡음이 있는 탐색 공간에서도 잘 작용한다는 점에서 강력한 도구이다 [Goldberg, 1989]. GA에 기초한 해법은 본 논문의 모델처럼 매우 복잡하고 비선형적인 비용 모델에 적용하기 쉽다. 셋째, GA는 최선의 해를 줄 뿐만 아니라, 일군의 바람직한 해를 도출해 준다. 최종 집단에 있는 해들은 설계 대안들의 영향에 대한 중요한 직관을 제공해 준다. 예를 들어서, 최종 집단에 있는 모든 해들이 특정 노드에 특정 파일을 저장한다면 설계자는 그 파일을 그 노드에 저장하는 것이 중요하다는 것을 당연히 확신하게 될 것이다.

우리의 분산 데이터 베이스 설계 알고리즘은 한 GA내에 다른 하나의 GA를 포함한다(March and Rho [1994]로부터 발췌되었고, 부록3에 요약되어져 있다.). 외부 GA(outer genetic algorithm)는 데이터 배분을 다루고 내부 GA(inner genetic algorithm)는 작업 배분을 다룬다. 겹쳐진(nested) 접근법은 데이터 배분

과 작업 배분 사이의 의존성을 조금 더 쉽게 다룰 수 있게 해 주기 때문에 표준적인 접근법에 비해 유리하다. 위에서 논의된 것처럼, 작업 배분의 실행 가능성은 데이터의 배분에 의존한다-- 각 검색 작업은 필요한 데이터를 저장하는 노드에 배분되어져야 한다. 표준적인 GA를 가지고 이러한 형태의 제약조건을 적용하기는 어렵다.

더 나아가서, 겹쳐진 접근법은 다른 작업 배분 모델들을 쉽게 통합하는 것을 가능하게 해준다. 다른 분산 데이터 베이스 관리 시스템은 다른 질의 처리 모델들을 사용하기 때문에(즉, 질의 최적화 기능들) 그러한 탄력성은 분산 데이터 베이스의 설계에 바람직하다.

GA는 C++로 작성되었고 Unix 환경에서 실행된다. 실행 시간은 문제의 크기(즉, 노드와 질의의 수)와 알고리즘 매개 변수들(집단 크기(pool size)와 반복의 수)에 의존한다. GA를 평가하기 위해 앞의 예제와 유사한 문제들을 총 운영비용의 최소화와 평균 응답 시간의 최소화를 목적으로 풀었다. GA는 Sun Sparc 20 워크스테이션에서 6시간 이내에 만족할 만한 해를 도출하였다. 이 때 외부 알고리즘의 반복의 수와 집단 크기는 각각 1000과 50 이었고, 내부 알고리즘은 각각 5000과 300 이었다.

이 장의 나머지 부분에서 해가 GA에서 어떻게 표현되는지를 간단하게 기술한다. 알고리즘의 자세한 내용은 [Rho, 1995]에 제시되어 진다. 외부 GA를 위한 해의 표현은 프래그먼트 배분을 나타낸다. 내부 GA를 위한 해의 표현은 작업 배분 모델에서의 4가지 유형의 의사 결정들을 나타내는 4가지 부분으로 구성되어 진다: (1) 복사본 선택, (2) 유익한 세미조인의 선정, (3) 조인의 순서, (4) 조인 노드의 선택. 그림3은 한 해의 표현을 보여준다. 표현의 각 부분은 아래에서 논의되어진다.

프래그먼트 배분은 각 프래그먼트별로 n비트(bit)의 집합들로서 표현되어 진다 (n은 네트워크에서 노드의 수임). 한 비트는 상응하는 파일 프래그먼트가 상응하는 노드에 배분되어 진다면 1의 값을 가지고 그렇지 않으면 0의 값을 가진다. 그림 3.a에서 보여지는 해(1110 1100 0010 ... 1000)는 본부, 지역1, 그리고 지역2에는 Customer1; 본부와 지역1에는 Account1; 지역2에는 Transaction1 등을 저장한다.

복사본 선택 결정은 질의에 의해 참조되어지는 각 프래그먼트를 위한 위치를 가진 벡터에 의해 표시되어 진다. 벡터의 각 위치의 값은 프래그먼트가 액세스되어지는 노드이다. 질의 R1.1은 Customer1, Account1, 그리고 Transaction1(그림2 참조) 프래그먼트를 필요로 한다. 본부에서 발생하는 이 질의에 대해 그림 3.b에서 보여지는 복사본 선택은 (복사본 선택 열에서 벡터(2 0 2)), 지역2로부터 Customer1, 본부로부터 Account1, 그리고 지역2로부터 Transaction1을 사용함을 명시한다.

세미조인 결정은 각 조인별로 2비트의 집합들로서 표현된다. 만약 세미조인이 수행된다면, 감축 파일(reducer)과 상응하는 비트의 값은 1이고 그렇지 않으면 0이다. 질의 R.1은 두개의 조인을 요구한다 (즉, 세개의 프래그먼트가 조인되어야 한다.). 본부에서 실행되는 이 질의를 위한 세미조인 결정은 (그림 3.b의 세미조인 열에서 비트 집합 (0110)) Account1 --> Customer1 그리고 Account1 --> Transaction1 세미조인의 실행을 명시한다.

조인 순서 결정은 배열이 조인이 수행되는 순서를 나타내는 조인의 목록으로서 나타난다. 본부에서 발생되는 질의 R1.1을 위한 조인 순서의 결정(그림3.b의 조인 순서열에서 목록 (1 2))은 Customer1과 Account1사이의 조인(즉, 질의에서의 첫째 조인)이 Transaction1과의 조인(질의에서의 두 번째 조인) 이전에 수행된다는 것을 나타낸다.

조인 노드 결정은 질의에서의 각 조인을 위한 위치를 가진 벡터로 표시된다. 그 벡터의 각 위치의 값은 조인이 수행되는 노드이다. 본부에서 발생되는 질의 R1.1을 위한 조인노드 결정(즉, 그림3.b의 조인 노드열에 있는 벡터(0 0))은 두 개의 조인이 모두 본부에서 수행됨을 나타낸다.

6. 결론 및 추후 연구 방향

본 연구에서는 통합된 방식으로 데이터 배분과 운영 전략을 다루는 포괄적인 분산 데이터 베이스 설계 모델을 개발했다. 이 모델은 데이터 복사, 동시성 통제 체제, 세미조인에 의한 데이터 감축, 그리고 조인 순서 결정을 포함한다. 이러한 문제를 효율적으로 풀기 위해서 겹쳐진 GA를 개발한다. 물론 GA는 최적해를 보장하지는 못하지만 예제에 대한 만족할 만한 해를 적절한 시간 내에 도출하였다.

앞으로의 연구를 위한 몇가지 영역이 있다. 첫째, 여러 분산데이터베이스 설계 모델을 비교 평가하는 연구가 이루어져야한다. 즉, 데이터와 작업 배분 전략이 분산 데이터 베이스 시스템의 효율성에 미치는 영향을 실제적인 기업 문제들을 사용하여 다양한 조건 아래에서 분석하여야 한다 (예를 들면, 다른 매개 변수를 가지는 다른 형태의 네트워크들). 둘째, 모델은 실제적인 환경에서 평가되고 검증되어야 한다. 선택된 해들은 실제 조직의 환경에서 수행되고, 그들의 성과가 측정되어야 한다. 셋째, 다양한 해법 알고리즘을 개발하고 평가하기 위해 많은 노력이 요구된다. 가능한 후보들은 Simulated Annealing, 부분적인 열거(partial enumeration) 기술, 그리고 라그랑제 완화법(Lagrangian relaxation) 등을 포함한다. 마지막으로, 모델 그 자체가 조금 더 현실적으로 개선될 수 있다. 가능한 개선으로는 데이터 이용 가능성(availability)의 모델화, 동적인 시스템 부하, 그리고 다른 처리 우선권 등이 있다.

參考 文獻

Apers, P. M. G., "Data Allocation in Distributed Database Systems," *ACM Transactions on Database Systems*, Vol. 13, No. 3, September 1988, pp. 263-304.

Apers, P. M. G., Hevner, A. R., and Yao, S. B., "Optimization Algorithms for Distributed Queries," *IEEE Transactions on Software Engineering*, Vol. SE-9, No. 1, January 1983, pp. 57-68.

Bernstein, P. A. and Chiu, D. W., "Using Semi-Joins to Solve Relational Queries," *Journal of the ACM*, Vol. 28, No. 1, January 1981, pp. 25-40.

Bernstein, P. A. and Goodman, N., "Concurrency Control in Distributed Database Systems," *ACM Computing Surveys*, Vol. 13, No. 2, June 1981, pp. 185-222.

Bernstein, P. A., Shipman, D. W., and Rothnie, J. B., Concurrency Control in a System for Distributed Databases (SDD-1), *ACM Transactions on Database Systems*, Vol. 5, No. 1, March 1980, pp. 18-51.

Blankinship, R., Hevner, A. R., and Yao, S. B., "An Iterative Method for Distributed Database Design," *Proceedings of the 17th International Conference on Very Large Data Bases*, Barcelona, Spain, September 1991, pp. 389-400.

Ceri, S. and Owicki, S., On the Use of Optimistic Methods for Concurrency Control in Distributed Databases, *Proceedings of 6th Berkeley Workshop on Distributed Data Management and Communication Networks*, Berkeley, CA, February 1982, pp. 117-130.

Ceri, S., Pernici, B., and Wiederhold, G., "Distributed Database Design Methodologies," *Proceedings of the IEEE*, Vol. 75, No. 5, May 1987, pp. 533-546.

Cornell, D. W. and Yu, P. S., "On Optimal Site Assignment for Relations in the Distributed Database Environment," *IEEE Transactions on Software Engineering*, Vol. 15, No. 8, August 1989, pp. 1004-1009.

Dowdy, L. W. and Foster,D. V., "Comparative Models of the File Assignment Problem," *ACM Computing Surveys*, Vol. 14, No. 2, June 1982, pp. 287-314.

Eswaran, K. P., "Placement of Records in a File and File Allocation in a Computer Network," in *Information Processing '74*, Stockholm, 1974, pp. 304-307.

Epstein, R., Stonebraker, M., and Wong, E., Query Processing in a Distributed Relational Database System, *Proceedings of ACM SIGMOD*, Austin, TX, May 1978.

Gardy, D. and Puech, C., "On the Effects of Join Operations on Relation Sizes," *ACM Transactions on Database Systems*, Vol. 14, No. 4, December 1989, pp. 574-603.

Gavish, B. and Sheng, O. R. L., "Dynamic File Migration in Distributed Computer Systems," *Communications of the ACM*, Vol. 33, No. 2., February 1990, pp. 177-189.

Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

Hevner, A. R., *The Optimization of Query Processing on Distributed Database Systems*, Ph.D. Thesis, Purdue University, 1979.

Hevner, A. R. and Yao, S. B., "Query Processing in Distributed Database Systems," *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 3, May 1979, pp. 177-187.

Kleinrock, L., *Queueing Systems: Theory*, John Wiley & Sons, 1975.

Lee, H. and Sheng, O. R. L., "A Multiple Criteria Model for the Allocation of Data Files in a Distributed Information Systems," *Computers and Operations Research*, Vol. 21, 1992, pp. 21-33.

Lohman, G. M., Mohan, C., Haas, L. M., Daniels, D., Lindsay, B. G., Selinger, P. G., and Wilms, P.F., "Query Processing in R*," in Kim, W. et al. (eds.) *Query Processing in Database Systems*, Spring-Verlag, Berlin, 1985, pp. 31-47.

March, S. T., "Techniques for Structuring Database Records," *ACM Computing Surveys*, Vol. 15, No. 1, March 1983, pp. 45-79.

March, S. T. and Rho, S., "Allocating Data and Operations to Nodes in Distributed Database Design," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 7, No. 2, April 1995, pp. 305-317.

Martin, T. P., Lam, K. H., and Russell, J. I., "Evaluation of Site Selection Algorithms for Distributed Query Processing," *Computer Journal*, Vol. 33, No. 1, February 1990, pp. 61-70.

Mishra, P. and Eich, M. H., "Join Processing in Relational Databases," *ACM Computing Surveys*, Vol. 24, No. 1, March 1992, pp. 63-113.

Mohan, C., Lindsay, B., and Obermarck, R., Transaction Management in the R* Distributed Database Management System, ACM Transactions on Database Systems, Vol. 11, No. 4, December 1986, pp. 378-396.

Navathe, S., Ceri, S., Wiederhold, G, and Dou, J., "Vertical Partitioning Algorithms for Database Design," *ACM Transactions on Database Systems*, Vol. 9, No. 4, December 1984, pp. 680-710.

Ozsu, M. and Valduriez, P., "Distributed Database Systems: Where Are We Now?" *IEEE Computer*, August 1991a, pp. 68-78.

Ozsu, M. and Valduriez, P., *Principles of Distributed Database Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1991b.

Ram, S. and Marsten, R. E., "A Model for Database Allocation Incorporating a Concurrency Control Mechanism," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 3, 1991, pp. 389-395.

Ram, S. and Narasimhan, S., "Allocation of Databases in a Distributed Database System," *Proceedings of the 11th International Conference on Information Systems*, December 1990, pp. 215-230.

Ram, S. and Narasimhan, S., "Database Allocation in a Distributed Environment: Incorporating a Concurrency Control Mechanism and Queuing Costs," *Management Science*, Vol. 40, No. 8, August 1994, pp. 969-983.

Rho, S., *Distributed Database Design: Allocation of Data and Operations to Nodes in Distributed Database Systems*, Unpublished Ph.D. Thesis, University of Minnesota, May 1995.

Rho, S. and March, S. T., "A Nested Genetic Algorithm for Distributed Database Design," *Proceeding of the 27th Hawaii International Conference on System Sciences*, January 1994, pp. 33-42

Ricciuti, M., "DBMS Vendors Chase Sybase for Client/Server," *Datamation*, Vol. 39, July 1, 1993, pp. 27-28.

Richter, J., "Distributing Data," Byte, June 1994, pp. 139-148.

The, L., "Distribute Data Without Choking the Net," *Datamation*, Vol. 40, January 7, 1994, pp. 35-36.

Yoo, H. and Lafortune, S., "An Intelligent Search Method for Query Optimization by Semijoins," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, No. 2, June 1989, pp. 226-237.

Yu, C. T. and Chang, C. C., "Distributed Query Processing," *ACM Computing Surveys*, Vol. 16, No. 4, December 1984, pp. 399-433.

Yu, C. T. and Chang, C. C.., "On the Design of a Distributed Query Processing algorithm," *Proceedings of the ACM-SIGMOD International Conference on the Management of Data*, San Jose, 1983, pp. 30-39.

Customer (10,000 레코드, 960,000 바이트)

c-id	문자	5
c-name	문자	20
ssn	문자	9
c-address	문자	30
c-city	문자	20
c-state	문자	2
c-zip	문자	10

Account(15,000 레코드, 1,350,000 바이트)

acc-no	문자	8
c-id	문자	5
br-id	문자	5
a-type	문자	2
a-status	문자	2
s-balance	수자	15.2
s-date	날짜	8
c-balance	수자	15.2
period-interest	수자	15.2
ytd-interest	수자	15.2

Transaction(300,000 레코드, 2,350,000 바이트)

t-id	문자	10
acc-no	문자	8
loc-id	문자	5
date	날짜	8
time	시간	8
amount	수자	15.2
t-type	문자	2
t-status	문자	2
t-ref	문자	20

그림 1. 예제 분산 데이터베이스 시스템의 테이블

a. 검색 질의

R1. 고객 보고서

```
SELECT  c-id, c-name, c-address, c-city, c-state, c-zip, acc-no, s-balance,
        c-balance, period-interest, ytd-interest, t-id, t-type, amount
FROM    Customer, Account, Transaction
WHERE   Customer.c-id = Account.c-id
AND     Account.acc-id = Transaction.acc-id
AND     Account.br-id = [지역]
```

R2. 잔고 조회

```
SELECT  c-id, c-name, acc-no, c-balance
FROM    Customer, Account
WHERE   Customer.c-id = Account.c-id
```

R3. 지점 보고서

```
SELECT  br-id, acc-no, c-balance
FROM    Account
WHERE   br-id = [지역]
AND     acc-no = [명시됨]
```

b. 갱신 질의

U1. 잔고 조정

```
UPDATE Account
SET     c-balance = [변경된 잔고]
WHERE   acc-no = [명시됨]
```

U2. 고객 자료 갱신

```
UPDATE Customer
SET     c-address = [명시됨], c-city = [명시됨], c-state = [명시됨],
        c-zip = [명시됨]
WHERE   c-id = [명시됨]
```

U3. 거래 기록

```
INSERT INTO   Transaction
VALUES ('t-id', ..... , 't-ref')
```

그림 **2.** 예제 분산 데이터베이스 시스템의 질의

a. 외부 알고리즘 해의 표현

프래그멘트	프래그멘트 배분
Customer 1	1110
Account 1	1100
Transaction 1	0010
Customer 2	1100
Account 2	1110
Transaction 2	0100
Customer 3	1000
Account 3	0001
Transaction 3	1000

b. 내부 알고리즘 해의 표현

질의	실행노드	복사본 선택	세미조인	조인순서	조인노드
R1.1	본부	2 0 2	01 10	1 2	0 0
R1.1	지역 1	2 1 2	01 10	1 2	1 1
R1.2	지역 2	1 2 1	01 10	1 2	2 2
R1.3	지역 3	0 3 0	01 10	1 2	3 3
R2.1	본부	0 0	00		0
R2.1	지역 1	1 1	00		1
R2.1	지역 2	2 1	00		2
R2.2	본부	0 0	00		0
R2.2	지역 1	1 1	00		1
R2.2	지역 2	1 2	01		2
R2.3	본부	0 3	00		0
R2.3	지역 3	0 3	01		3
R3.1	본부	0			
R3.1	지역 1	1			
R3.2	본부	0			
R3.2	지역 2	2			
R3.3	지역 3	3			

그림 3. GA에서의 해의 표현

부록 1. 총 운영 비용

1. $\mathbf{COM(k,m) = \sum_{t}\sum_{p\neq t} H(k,m,t,p)\, c_{tp}}$

여기서 c_{tp} 는 노드 t에서 p로의 바이트 당 전송 비용이다.

검색 메시지 단계,

$H(k,m,t,p) = L^M$ if t = node(k) and p = node(a(k,m))

$H(k,m,t,p) = 0$ otherwise

L^M 은 메시지의 크기이고, node(k) 는 질의 k의 실행 노드이며, node(i) 는 파일 프래그멘트가 저장되어 있는 노드이다.

조인 단계,

$H(k,m,t,p) = L_{a(k,m)} + L_{b(k,m)}$ if t = node(a(k,m)) = node(b(k,m)) and p = node(k,m)

$H(k,m,t,p) = L_{a(k,m)}$ if t = node(a(k,m)) and t ≠ node(b(k,m)) and p = node(k, m)

$H(k,m,t,p) = L_{b(k,m)}$ if t ≠ node(a(k,m)) and t = node(b(k,m)) and p = node(k, m)

$H(k,m,t,p) = 0$ otherwise.

L_i 는 파일 프래그멘트 i의 크기, a(k,m)과 b(k,m)은 질의 k의 단계 m에 의해 참조되어지는 파일 프래그멘트, 그리고 node(k,m) 는 질의 k의 단계 m이 수행되는 노드이다.

최종 단계,

$H(k,m,t,p) = L_{a(k,m)}$ if t = node(a(k,m)) and p = node(k)

$H(k,m,t,p) = 0$ otherwise.

갱신의 보내는 메시지 단계,

$H(k,m,t,p) = L^M$ if t = node(k) and copy(a(k,m), p) = 1

$H(k,m,t,p) = 0$ otherwise

copy(i,t)는 프래그멘트 i가 노드 t에 저장되면 1 그렇지 않으면 0이다.

갱신의 받는 메시지 단계,

$H(k,m,t,p) = L^M$ if copy(a(k,m), t) = 1 and p = node(k)

$H(k,m,t,p) = 0$ otherwise.

2. $\mathbf{IO(k, m) = \sum_{t} O(k,m,t)\, d_t}$

d_t 는 노드 t에서의 디스크 입출력 단위 비용이다.

선택 및 프로젝션 단계,

$O(k,m,t) = D_{kmt}$ if t = node(a(k,m))

O(k,m,t) = 0 otherwise

D_{kmt} 는 노드 t에서 질의 k의 단계 m을 처리하는 데 필요한 디스크 입출력 수이다.

조인 단계,

O(k,m,t) = $F_{a(k,m)t}$ if t ≠ node(k, m) and t = node(a(k,m)) and t ≠ node(b(k,m))
O(k,m,t) = $F_{b(k,m)t}$ if t ≠ node(k, m) and t ≠ node(a(k,m)) and t = node(b(k,m))
O(k,m,t) = $F_{a(k,m)t} + F_{b(k,m)t}$ if t ≠ node(k, m) and t = node(a(k,m)) and t = node(b(k,m))
O(k,m,t) = D_{kmt} if t = node(k,m) = node(a(k,m)) = node(b(k,m))
O(k,m,t) = $D_{kmt} + E_{a(k,m)t}$ if t = node(k,m) = node(b(k,m)) and t ≠ node(a(k,m))
O(k,m,t) = $D_{kmt} + E_{b(k,m)t}$ if t = node(k, m) = node(a(k,m)) and t ≠ node(b(k,m))
O(k,m,t) = $D_{kmt} + E_{a(k,m)t} + E_{b(k,m)t}$ if t = node(k,m) and t ≠ node(a(k,m)) and t ≠ node(b(k,m))
O(k,m,t) = 0 otherwise

$F_{a(k,m)t}$ 는 a(k,m)을 노드 t에서 다른 노드로 보내는 데 필요한 디스크 입출력 수이고, $E_{a(k,m)t}$ 는 a(k,m)을 노드 t에서 받아 저장하는 데 필요한 디스크 입출력 수이다.

최종 단계,

O(k,m,t) = $E_{a(k,m)t}$ if t ≠ node(a(k,m)) and t = node(k)
O(k,m,t) = $F_{a(k,m)t}$ if t = node(a(k,m)) and t ≠ node(k)
O(k,m,t) = 0 otherwise.

갱신 단계,

O(k,m,t) = D_{kmt} if copy(a(k,m), t) = 1
O(k,m,t) = 0 otherwise

3. CPU(k, m) = $\sum_{t}$ U(k,m,t) p_t

p_t 는 중앙처리장치의 단위 비용이다.

메시지 단계,

U(k,m,t) = S_t if t = node(k) and t ≠ node(a(k,m))
U(k,m,t) = R_t if t ≠ node(k) and t = node(a(k,m))
U(k,m,t) = 0

S_t 와 R_t 는 각각 메시지를 보내고 받는 데 .

선택 및 프로젝션 단계,

U(k,m,t) = W_{kmt} if t = node(a(k,m))
U(k,m,t) = 0 otherwise

W_{kmt} 는 노드 t에서 질의 k의 단계 m을 처리하는 데 필요한 중앙처리장치 조작 수이다

조인 단계,

$U(k,m,t) = F'_{a(k,m)t}$ if $t \neq node(k, m)$ and $t = node(a(k,m))$ and $t \neq node(b(k,m))$
$U(k,m,t) = F'_{b(k,m)t}$ if $t \neq node(k, m)$ and $t \neq node(a(k,m))$ and $t = node(b(k,m))$
$U(k,m,t) = F'_{a(k,m)t} + F'_{b(k,m)t}$ if $t \neq node(k, m)$ and $t = node(a(k,m))$ and $t = node(b(k,m))$
$U(k,m,t) = W_{kmt}$ if $t = node(k, m) = node(a(k,m)) = node(b(k,m))$
$U(k,m,t) = W_{kmt} + E'_{a(k,m)t}$ if $t = node(k, m) = node(b(k,m))$ and $t \neq node(a(k,m))$
$U(k,m,t) = W_{kmt} + E'_{b(k,m)t}$ if $t = node(k, m) = node(a(k,m))$ and $t \neq node(b(k,m))$
$U(k,m,t) = W_{kmt} + E'_{a(k,m)t} + E'_{b(k,m)t}$ if $t = node(k, m)$ and $t \neq node(a(k,m))$ and $t \neq node(b(k,m))$
$U(k,m,t) = 0$ otherwise

$F'_{a(k,m)t}$ 는 a(k,m)을 노드 t에서 다른 노드로 보내는 데 필요한 중앙처리장치 조작 수이고, $E'_{a(k,m)t}$ 는 a(k,m)을 노드 t에서 받아 저장하는 데 필요한 중앙처리장치 조작 수이다.

최종 단계,

$U(k,m,t) = E'_{a(k,m)t}$ if $t \neq node(a(k,m))$ and $t = node(k)$
$U(k,m,t) = F'_{a(k,m)t}$ if $t = node(a(k,m))$ and $t \neq node(k)$
$U(k,m,t) = 0$ otherwise.

갱신의 보내는 메시지 단계,

$U(k,m,t) = \sum_{p \neq t} copy(a(k,m), p)\, S_t$ if $t = node(k)$
$U(k,m,t) = R_t$ if $t \neq node(k)$ and $copy(a(k,m), t) = 1$
$U(k,m,t) = 0$ otherwise

갱신의 받는 메시지 단계,

$U(k,m,t) = \sum_{p \neq t} copy(a(k,m), p)\, R_t$ if $t = node(k)$
$U(k,m,t) = S_t$ if $t \neq node(k)$ and $copy(a(k,m), t) = 1$
$U(k,m,t) = 0$ otherwise

갱신 단계,

$U(k,m,t) = W_{kmt}$ if $copy(a(k,m), t) = 1$
$U(k,m,t) = 0$ otherwise

4. $\mathbf{STO(t) = s_t \sum_i copy(i, t)\, L_i}$

s_t 는 노드 t에서의 단위 시간당 단위 저장 비용이다.

부록 2. 평균 응답 시간

1. $\mathbf{R_{COM}(k)} = \sum_t \sum_p \sum_m \left(\frac{W(t,p)TL(t,p)N(k,m,t,p)}{(UL(t,p))^2 - UL(t,p)TL(t,p)} + \frac{H(k,m,t,p)}{UL(t,p)} \right)$

UL(t,p) 는 노드 t에서 p로의 통신 링크의 용량(단위 시간당 전송 가능 바이트 수)이고,

$TL(t,p) = \sum_k f(k) \sum_m H(k,m,t,p)$, $W(t,p) = \frac{TL(t,p)}{\sum_k f(k) \sum_m N(k,m,t,p)}$, 그리고 N(k,m,t,p) 는

$H(k,m,t,p) > 0$ 이면 1이고 그렇지 않으면 0이다.

2. $\mathbf{R_{IO}(k)} = \sum_t \sum_m O(k,m,t) \frac{1}{UIO(t) - TIO(t)}$

UIO(t) 는 노드 t에서의 디스크 입출력 용량(단위 시간당 디스크 입출력 수)이고

$TIO(t) = \sum_k f(k) \sum_m O(k,m,t)$ 는 노드 t에서의 총 디스크 입출력 수이다.

3. $\mathbf{R_{CPU}(k)} = \sum_t \sum_m U(k,m,t) \frac{1}{UCPU(t) - TCPU(t)}$

UCPU(t) 노드 t에서의 중앙처리장치 용량(단위 시간당 중앙처리장치 조작 수)이고

$TCPU(t) = \sum_k f(k) \sum_m O(k,m,t)$ 는 노드 t에서의 총 중앙처리장치 조작 수이다.