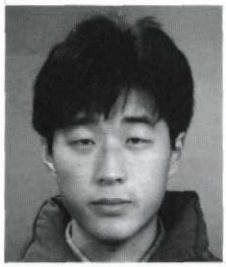
고성능 데이터베이스시스템 : 분산/병렬 DBMS기술

High-performance Database System:Dispersion/Parallel DBMS Technology



김용호
한국기계연구원 지역정보유통망사업단 연구원
Kim yong-ho./ Korea Institute of Machinery and Metals Reginal Info. Network for Sci-tech.
Researcher.

1. 분산/병렬 DBMS기술의 소개

십 오년전쯤 병렬 데이터베이스기술이 나오기 시작할 때만 하더라도 별로 호응을 받지 못했다. 그 당시 대부분의 데이터베이스 머쉰들이 전하결합소자기억장치(CCD:Charge Coupled Device)메모리, 버블 메모리, 헤드-당-헤드 (Head-per-Head)디스크, 그리고 광디스크들을 이용하였다.

이러한 환경은 하드웨어의 속도가 느리기 때문에 데이터베이스 처리를 위해서는 만족할만한 환경이 아니었다. 하지만 하드웨어 기술이 발달하면서 CPU, RAM, 그리고 대용량의 하드 디스크들이 화일 처리 방식이 아닌 데이터베이스 처리 방식이 발달하기 시작했다.

그러나 CPU, RAM, 그리고 통신망의 속도는 급격히 빨라지는데 비해 기계적인 장치인 디스크의 속도는 상대적으로 너무 느렸다. 특히 데이터베이스시스템에서 볼때에는 디스크 I/O 시간이 컴퓨터 성능의 대부분을 차지하게 되었다.

따라서, 데이터베이스기술에서 디스크에 데이터를 읽고 쓰는 시간을 최대한 줄이는 것만이 매우 중요한 문제로 떠올랐다.

그것을 해결하기 위해, 다중 처리기 시스템이나 고속통신망 상에 분산된 노드들에게 데이터를 분배 또는 분산시킨 다음에 동시에 읽어서, 각 노드에서 또는 노드끼리 협력하도록 하면서, 사용자의 질의를 처리하게 하는 분산/병렬 데이터베이스기술이 하나의 해결 방안으로 떠오르게 되었다.

이러한 기술은 하드웨어와 함께 꾸준한 연구가 있었으며, 상용 시스템으로는 테라테이터,

탠덤 등과 같은 초병렬 시스템이 잘 알려져 있다.

여기서 왜 분산/병렬 데이터베이스시스템이 가능한가를 잠시 살펴 보겠다.

병렬 데이터베이스시스템은 기존의 관계형 데이터베이스를 폭넓게 수용한다. 많은 관계형 질의문들이 병렬 수행 가능하고, 그런 질의문에 의해 병렬 디스크에 분산된 데이터들을 동시에 읽어 낼 수 있다. 그리고 관계형 데이터베이스의 데이터 흐름을 데이터 플로우(Data Flow) 방식으로 접근하여 표현할 수 있다.

그것은 관계형 데이터베이스 자체에 병렬성이 존재한다는 것을 말한다. 즉, 데이터베이스연산들을 수행하는 프로세스들이 하나의 처리 단위가 되고, 그러한 프로세서들이 서로 통신망에 서로 연결되어 있어서 메세지에 의한 클라인어트/서버(Message-Based Client/Server) 구조로 서로 협력하도록 되어 있다. 디스크에서 읽혀진 데이터들은 그러한 프로세서에서 처리되어 사용자가 요구한 결과로 가공되어지는 것이다.

클라이언트/서버 패러다임은 고속의 LAN환경에서 연결된 PC나 워크스테이션들을 기본으로 하는데, 이것은 곧 분산/병렬 처리 방식이 초고속 통신망을 효율적으로 이용하기 위해서 필요로 하는 기술임을 알 수 있다.

구조적으로 보면 분산/병렬 처리 방식은 클라이언트/서버 구조와 거의 같음을 알 수 있다. 그런 의미에서 클라이언트/서버 패러다임은 새로운 개념이 아니라 통신망 환경이 개선됨에 따라 과거의 분산/병렬 환경을 재조명하고, 현재의 환경에 맞게 정리된 개념이라고 볼 수 있다.

이미 대용량의 메인프레임 설계자들은 RDBMS에서 많은 수의 동시 사용자들을 수용하거나 테라바이트 단위의 데이터베이스를 처리할 수 있도록 CPU와 I/O 요구사항을 동시에 만족하기에는 어렵다는 것을 알고 있다.

그러한 동안에 빠르고 값싼 마이크로프로세서들을 이용한 Encore, Intel, NCR, nCUBE, Sequent, Tandem, Teradata 등의 다중 프로세서 컴퓨터들이 밴더(Vendor)들을 통해 널리 보급되고 있다.

여태까지 분산/병렬 DBMS의 대상은 RDB인데, 그 이유는 RDBMS는 그 자체가 병렬성을 가지고 있기 때문에 쉽게 적용할 수 있었기 때문이다. 크게 두가지로 나누어 볼 수 있다.

첫째는, 데이터들이 각각의 데이터 집합으로 쉽게 나눌 수 있다는 것(Data Partitioning)이다.

RDB의 데이터들을 영역분배(Range Partition), 라운드로빈분배(Round Robin Partition), 해쉬분배 (Hash Partition)방법 등을 통해서 분산되어 있는 노드에 데이터를 분산하여 저장하고 관리하기가 수월하다.

또 하나는 관계연산 자체가 병렬성을 내포하고 있다는 것 (Parallelism Within Relational Operation)이다. 질의문에 의한 관계연산이 주어질 때, 데이터가 분산시스템이나 다중프로세서시스템의 각 노드에 분산되어 있을 때, 각 노드마다 필요한 형태의 관계 연산을 적용시켜 동시에 데이터를 액세스해서 데이터를 읽거나 쓸 수 있다.

2장에서는 기존에 연구된 분산/병렬 시스템의 사례를 살펴 보고, 3장에서는 앞으로의 방향에 대해서 나름대로 잠깐 언급하겠다.

2. 사례 연구

현재까지 다중프로세서시스템을 이용한 시스템들에 간략하게 살펴 보면, Teradata, Tandem Nonstop SQL, Gamma, SDC(Super Database Computer), Bubba, XPRS, Volcano 등

등이 있다.

이중에는 상용화된 것도 있고 대학에서 연구를 목적으로 개발된 시스템이 있는데, 각각에 대해 특징을 살펴 보면 아래와 같다.

Teradata는 보통의 컴퓨터 상에서 동작하는 클라이언트 프로그램에 대해서 SQL서버가 있는 형태를 갖추고 있으며, 1,000개 이상의 프로세서와 몇 천개의 디스크를 관리할 수 있다.

프로세서는 IFP(Interface Processor)들과 AMP (Access Module Processor)들로 나누어져 있고, 내부적으로 Y-net으로 각각 연결되어 있다.

각각의 릴레이션(Relation)들은 AMP상에 해쉬에 의해 분배되어 있다. Teradata는 여러 시스템에서 구축되었는데, 좋은 성능을 보여 주었고, 대용량의 데이터베이스를 관리하는데에 있어서 메인프레임보다 훨씬 나은 성능을 보인다.

Tandem, NonStop, SQL은 4-Plexed Fiber Optic Ring에 프로세서 클러스터 (Cluster)들이 연결되어 구성된 시스템이다.

다른 시스템과는 달리 데이터베이스 서버로써 O/S를 구동시키고, 같은 프로세서들에서 응용 프로그램을 수행하기 때문에, 프로그램과 머쉰간에 Front-end와 Back-end가 구분되지 않는다.

또 시스템 면에서 특징은 1MIPS 에 하나의 디스크가 붙도록 되어 있다 (예를 들면, 10MIPS 에는 열개의 디스크가 연결된다). 릴레이션들은 라운드로빈 방법으로 다중 디스크상에 분배되기가 쉽고, B-tree 구조는 이차 색인들을 구성하기 위해 지원한다.

Tandem은 주로 OLTP시스템으로 설계되어 있다.

OLTP에서 병렬성은 색인들을 고칠때(Index Update), 병렬적으로 할 수 있다는 것이다.

SQL릴레이션은 보통 다섯개의 색인이 있다. 이러한 색인들은 데이터를 부분적으로 읽는 속도를 증가시키는데는 유리하지만, 데이터 삽입(Insert), 수정(Update)에는 속도가 느리다. OLTP응용에서 전체적으로는 메인 프레임보다는 월등하다고 평가한다.

Gamma는 위스콘신 대학에서 만든 프로토타입(Prototype) 데이터베이스 시스템으로써, 각 노드에 디스크가 붙어 있는 iPSC/2 하이퍼 큐브 상에서 DBMS를 구현한 것이다.

각 노드마다 WISS라는 저장 시스템과 NOS가 있어서 노드끼리 통신하고 데이터를 주고 받는 형태이다. 색인은 B-tree나 해쉬를 이용한다. 또 병렬알고리즘과 파이프라인 두가지 방식을 사용해서 관계 연산을 수행한다. 죠인은 세가지 방식의 해쉬 죠인 알고리즘과 정렬병합(Sort Merge)죠인 알고리즘을 사용한다.

SDC(Super Database Computer) 프로젝트는 동경 대학에서 추진한 것으로 다른 시스템과 비교가 된다. 하드웨어적인 부분과 소프트웨어적인 부분을 서로 묶어 놓았다는 점에서 비교가 되는데, PM (Process Module)이라는 기본 단위들을 내부적으로 NxN으로 연결하는 형태의 오메가 (Omega)내부 네트웍을 사용한다. 데이터가 오메가 네트웍을 통해 PM을 거치면 결과적으로는 데이터들이 자동적으로 정렬이 되도록 하는 독특한 구조이다.

Bubba는 40개의 디스크를 가지는 40-node ELEX/32 다중 프로세서 시스템을 사용해서 구현한 것이다.

이 시스템이 메모리공유(Shared Memory) 방식을 지원하지만 Bubba는 무공유(Shared Nothing)형태의 접근 방식으로 설계되어졌다. 메모리 공유 방식으로 구현된 것은 단지 메세지

전달(Message Passing)방법만을 이용하였다. 노드들은 세개의 그룹으로 나누어져 있는데, 하나는 외부의 호스트 프로세서들과 통신하기 위한 인터페이스프로세서(Interface Processor)이고, 다음은 데이터 저장과 질의수행을 지능적으로 관리하는 등위질의수행기(Coordinating Query Execution)이고, 마지막 하나는 체크포인트/로깅(CheckPoint/Logging)부분이다.

Bubba는 저장 메카니즘으로 데이터분배방식을 택하고, 데이터 플로우 메카니즘을 따른다. Bubba는 특별한 몇가지 방법을 사용하는데, 하나는 인터페이스 언어로써 SQL에 비해 퍼시스턴스(Persistence)를 유지할 수 있는 확장된 관계형 언어인 FAD를 사용한다.

또 하나는 데이터 저장 메카니즘이 각 노드에서 수행되는 프로세스의 가상 메모리 주소 공간을 단일 레벨(single-level)로 저장하는 것이다. 이러한 것이 전형적인 구축 방법과 다르다.

다른 종류의 시스템으로는 XPRS, Volcano, Arbre, 그리고 PERSIST등이 있다. Volcano와 XPRS 는 메모리 공유 방식을 택하고 있다.

그외 Oracle도 64-node nCUBE무공유 방식의 시스템에서 병렬 DBMS 를 만들었고, 성능은 TPC-B벤치마크에서 기존의 Oracle머쉰보다 훨씬 뛰어난 성능을 보여 주었다.

3. 앞으로의 방향

병렬 DBMS는 OLTP와 같은 특정한 응용에서는 상당히 좋은 평가를 받고 있다. 하지만 범용 시스템의 성격을 지니기에는 분산/병렬이라는 자체가 기술적인 면에서 사실상 어려움이 많다.

예를 들어, 하나의 큰 릴레이션을 처리하는데 데이터들이 분산되어 있기 때문에 잠금 (Locking)을 하는데 상대적으로 많은 시간이 소요될 것임을 알 수 있다.

우선 순위 스케쥴링(Priority scheduling)을 하기위해서는 분산된 노드들간에 복잡한 메카니즘이 요구된다. 그리고 병렬 질의들을 최적화하기도 그리 만만치가 않음을 쉽게 생각해 볼 수 있다.

하지만 이것은 꼭 병렬 시스템에만 적용되는 것은 아니다. 다만 병렬 시스템에서는 좀더 복잡할 뿐이라는 것을 말하고 싶다.

분산/병렬 데이터베이스의 데이터 분배성(Data Partitioning), 데이터 플로우식 접근 방법, 그리고 내재하는 병렬성(Intraparallelism)등을 잘 이해하면, 이미 존재하는 DBMS들을 고성능의 병렬 DBMS로 구축할 수 있다. 그러면 대용량의 I/O자원과 연산 시간을 요구하는 특별한 응용 (예를들면, Terabyte단위의 데이터베이스를 처리하는 응용) 에서는 병렬 시스템을 이용해서 아주 큰 효과를 얻을 수 있다.

컴퓨터 통신기술이 급격히 나아짐에 따라 통신망에 분산된 시스템들을 하나의 시스템으로 이용하는 기술이 이미 일반화되고 있고, 또 멀티미디어와 같은 대용량의 데이터들을 상대적으로 늦은 I/O장치들을 이용한다는 점에서 분산/병렬 처리는 앞으로 불가피하다.

어떤 응용에서는 관계형 모델이 적용되지 않는다. 그것은 객체지향 데이터베이스시스템이라는 새로운 클래스를 형성하고 있다.

현재 관계형 모델에 객체지향형 모델을 추가한 UniSQL같은 시스템이 많은 관심을 끌고 있다. 하지만 이러한 형태의 시스템에서도 분산처리의 필요성을 시사한 바 있다.

시간이 갈수록 좀 더 안정된 분산/병렬 처리 기술이 요구될 것이고, 또 새로운 해결 방안들이 제시될 것이다. DC