Oracle9*i* Real Application Clusters와 IBM DB2 UDB EEE v7.2의 기술적 비교

오라클 기술 백서

2002년 2월

ORACLE

Oracle9*i* Real Application Clusters와 IBM DB2 UDB EEE v7.2의 기술적 비교

그림 목록

표 목록

Oracle9*i* Real Application Clusters와 IBM DB2 UDB EEE v7.2의 기술적 비교

개요

요즘은 현재의 경제적 풍토를 기초로 하든 비즈니스의 경쟁적 특성에 상관 없이, 글로벌한 기업 정보를 통합하기 위한 경향이 새로이 나타나고 있다. 이러한 기업 정보의 통합을 통해 회사들은 비용을 크게 절감하며, 세계적인 경영을 이끌어 나가는 데 필요한 서버 및 개인용 컴퓨터의 수를 줄일 수 있다(*"십억 달러의 비용 절감 방법"* 참조). 이러한 통합에서 얻을 수 있는 또 하나의 이점은 인터넷 상의 글로벌 데이타베이스에 모든 정보를 공유함으로써 모든 사람이 필요 시에 정보를 찾을 수 있는 위치를 알고 있다는 점이다. 이러한 중앙화된 데이타 공유 방식을 사용하면 끊임없이 증가하고 있는 내부 및 외부 사용자 커뮤니티의 수에 대해 협력적인 애플리케이션을 보다 쉽게 구축할 수 있다.

그러나 글로벌 데이타, 신뢰성, 가용성 및 확장성만으로는 더 이상 경쟁 상의 이점을 누릴 수 없으며, 이러한 요소들은 기본적인 요구 사항에 불과할 뿐이다. 이러한 요구 사항을 충족시켜야 할 IT 관계자에게 부과되는 부담은 매우 크며, 관리 체인 상위로 올라갈수록 그 압력이 증가하고 있다.

오늘날의 기술 관리자는 애플리케이션 및 데이타베이스 업그레이드를 계획 및 구입할 때 선택의 폭이 적기 때문에 훨씬 더 신중히 검토해야 한다. 그러나 이는 단순히 기술적인 결정이 아닌 비즈니스 상의 결정이다. ROI 계산은 전보다 훨씬 더 복잡하다. 가격 대비 성능은 IT 결정자에게 있어 매우 중요시되는 결정 요소이지만, 전략가들은 가격이 상대적이므로 시스템 업그레이드를 평가할 때 기타 여러 요소들도 고려해야 한다는 사실을 인지하고 있다.

- 변경이 필요할 때 리소스를 재배치할 수 있도록 플랫폼 전략이 유연성을 갖추고 있는가?
- 업그레이드가 일일 작업 환경 비용에 영향을 미칠 것인가?
- 기존 애플리케이션 또는 새 애플리케이션을 배치하는 데 얼마나 큰 어려움이 따를 것인가?

---

1 Ellison, Larry, Oracle, How We Saved A Billion Dollars, 2000

- 애플리케이션 아키텍처(OLTP, DSS)가 데이타베이스 성능에 어떠한 영향을 미칠 것인가?
- 관리자가 작업을 지원할 수 있으려면 얼만큼의 재교육이 필요한가?

선택할 수 있는 실질적인 동종의 데이타베이스 제품은 Oracle9*i* Real Application Clusters와 IBM의 DB2 Universal Database EEE V7.2, 이 두 가지 뿐이며, 선택은 간단할 것이다.

소개

이 백서는 Oracle9*i* Real Application Clusters(RAC) 환경과 비슷한 종류의 데이타베이스 제품인 IBM의 DB2 Universal Database EEE V7.2(DB2 EEE)에 실제 애플리케이션 배치하는 작업을 비교하는 데 역점을 두고 있다. 이러한 점에 역점을 두는 이유는 두 가지다. 첫째, 두 데이타베이스 제품 모두 많은 기능을 갖추고 있어 두 제품의 경쟁 범위로 이 백서의 범위를 제한해야 하기 때문이다. 둘째, 모듈식 성장을 통해 오늘날의 IT 환경에서 요구하는 신뢰성, 가용성 및 확장성과 같은 요건들을 충족시키려는 경우 클러스터링만이 비용 절감 방법을 제공할 유일한 솔루션이기 때문이다.

먼저 하드웨어 및 소프트웨어 플랫폼을 살펴보고 얼마나 우수한 제품인지, 특수한 요구 사항이 요구되는 경우 그 사항들은 무엇인지, 또한 그와 관련된 비용은 얼마인지 알아볼 것이다. 그런 다음, 애플리케이션의 배치와 개발 시스템을 구현하는 데 필요한 요소들을 살펴볼 것이다. 애플리케이션이 가동되고 실행되고 나면 데이타 센터에서 성능이 가장 중요시된다. 데이타베이스 애플리케이션마다 서로 다른 요구 사항 및 해결해야 할 특성들을 갖추고 있으므로 여기서는 이러한 애플리케이션을 OLTP, DSS 및 패키지 애플리케이션이라는 세 가지 환경으로 나눌 것이다.

먼저 하드웨어 및 소프트웨어 요구 사항이 비슷하다는 사실을 지적할 수 있다. 애플리케이션에 대한 추상화는 두 가지 솔루션 모두에서 동일하다. 즉, 단일 데이타베이스처럼 보이게 하며 SQL 코드를 수정할 필요가 없다. 배치 용이성, 성능 및 관리성 면에서 차이가 있다.

먼저, 사용할 정의와 용어를 분명히 하자.

DB2 용어에서 있어 분할(partition)이라는 용어는 논리적 노드와 논리적 노드에서 소유하는 데이타를 참조하는 데 사용되며, 테이블의 일부 또는 인덱스를 참조하는 RAC 사용과는 다르다. 이 문서에서는 "분할"이라는 단어를 사용하여 논리적 노드(예를 들면 DB2 사용)를 표현할 것이다. 각 분할마다 고유 버퍼 풀, 패키지 캐시 등이 있다. 데이타의 부분 집합에만 직접 액세스할 수 있다는 점을 제외하고는 분할을 Oracle 인스턴스처럼 생각하면 편리할 것이다.

하드웨어 요구 사항

두 개 데이타베이스는 클러스터 하드웨어 요구 사항이 비슷하다. 클러스터는 단일 시스템으로 협력하는 개별 서버의 그룹으로, 주 클러스터 구성 요소는 프로세서 노드, 클러스터 상호 연결(사설 네트워크) 및 디스크 하위 시스템이다.

클러스터는 디스크 액세스와 데이타를 관리하는 리소스를 공유하지만 각 개별 하드웨어 클러스터 노드는 메모리를 공유하지 않는다. 따라서 그림 1에서처럼 각 노드는 고유 전용 시스템 메모리와 고유 운영 체제, 데이타베이스 인스턴스 및 애플리케이션 소프트웨어를 갖추고 있다.

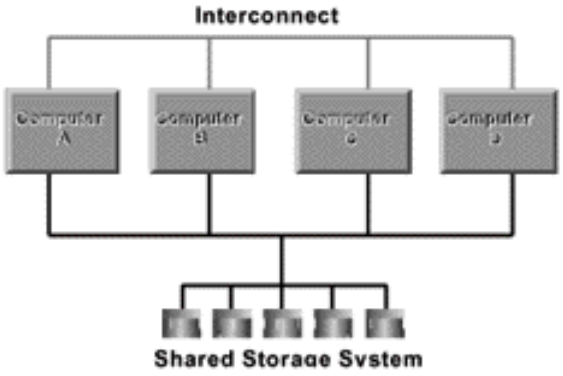

그림 1: 프로세서 노드, 클러스터 상호 연결 및 디스크 하위 시스템으로 구성된 클러스터

클러스터는 단일 대칭 멀티 프로세서 시스템 상에서 향상된 결함 회복력 및 모듈식 증분 시스템 성장을 제공할 수 있다. 하위 시스템 실패의 경우 클러스터링은 고 가용성을 유지한다. 추가 노드, 상호 연결 및 공유 디스크와 같은 중복 하드웨어 구성 요소는 보다 높은 가용성을 제공하는데, 이러한 중복 하드웨어 아키텍처는 단일 실패 지점을 피할 수 있으며 예외적인 결함 회복력을 제공한다.

개념적인 면에서 RAC와 DB2에 대한 클러스터 요구 사항은 비슷하다. 특히 고 가용성(HA)이 기본적인 요구 사항일 경우에는 더욱 그러하다. 클러스터에서 각 노드의 CPU 및 메모리 요구 사항은 단일 시스템 요구 사항과 비슷하며 이 설명서에서는 이 점을 별도로 구별하지는 않았다. 그러나 다른 클러스터 구성 요소 요구 사항에 대해서는 논의할 점들이 많은데, 특히 성능 및 비용에서 그러하며 이와 관련된 두 가지 요구 사항은 다음과 같다.

- 클러스터 상호 연결
- 공유 디스크 대 공유되지 않은 저장소

클러스터 상호 연결

클러스터의 각 노드는 다른 노드에게 상태와 구성 정보를 항상 알려주어야 한다. 이 작업은 네트워크 상에서 heartbeat라고 하는 네트워크 메시지를 브로드캐스트하여 정기적으로 수행된다. heartbeat 신호는 개별 상호 연결 상에서 일반적으로 전송되며 노드 간 통신에 사용된다.

이 클러스터 상호 연결은 각 노드에 네트워크 카드를 설치한 후 적당한 네트워크 케이블을 통해 카드를 연결한 다음 와이어 상에서 실행되도록 소프트웨어 프로토콜을 구성하는 방식으로 구축된다.

이 상호 연결은 가격 성능 곡선 상에서의 원하는 위치에 따라 HMP(Hyper Messaging Protocol)을 사용하여 RDG(Reliable DataGram) 또는 Hewlett-Packard의 Hyperfabric/2를 실행하는 Compaq의 메모리 채널과 같은 고속 독점 상호 연결 또는 TCP/IP나 UDP를 실행하는 저렴한 이더넷 카드일 수 있다. 그림 1은 일반 4-노드 클러스터 구성을 보여준다.

낮은 대기 시간/고속 상호 연결은 OLTP 애플리케이션에서의 캐시 전송 요구 사항으로 인한 RAC 성능과 DSS 애플리케이션의 병렬 질의 통신에 가장 적합하다. 또한 DB2는 OLTP 및 DSS 애플리케이션에서 조정자/작업자 프로세스 통신의 최고 성능을 위해 고속 상호 연결을 필요로 한다.

공유 디스크와 비공유 저장소

이 점은 아마도 RAC와 DB2 EEE 간의 가장 큰 차이점 중 하나일 것이다. 이러한 두 개 데이타베이스 아키텍처에 대한 많은 자료들이 있지만 여전히 많은 혼동이 있는 듯 하다. 이 두 가지 제품에 대한 개념과 구현 방법을 이해해야 하는 것은 매우 중요한데, 이는 애플리케이션을 배치할 때 총 소유 비용(TCO)에 큰 영향을 미치기 때문이다.

기존 비공유

순수 비공유 아키텍처에서 데이타베이스 파일은 다중 컴퓨터 시스템의 노드 상에서 실행되는 여러 인스턴스 사이에서 분할된다. 각 인스턴스 또는 노드는 데이타의 개별 부분 집합의 소유권을 가지고 있으며, 이 데이타에 대한 모든 액세스는 이 "소유" 인스턴스에서 배타적으로 수행한다. 다른 말로 하면 순수 비공유 시스템에서 분할되거나 제한된 액세스 방법을 사용하여 다수의 프로세싱 노드들 간에 작업을 분담한다. 이 작업은 노드의 데이타 소유가 거의 잘 변경되지 않는 환경에 적합하며, 소유권이 변경되는 일반적인 이유는 데이타베이스 재구성 또는 장애 때문이다.

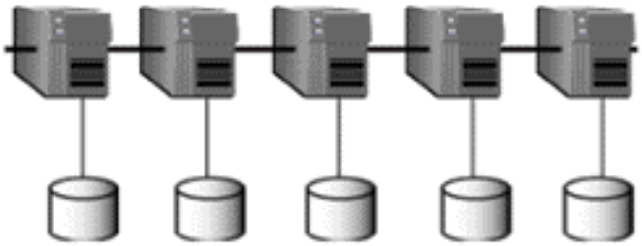

그림 2 : 비공유 클러스터 데이타베이스

비공유 시스템에서 병렬 실행은 데이타 분할 방법에 직접 기초하고 있다. 데이타가 정확히 분할되는 경우 시스템은 거의 직선 형태로 확장된다.

이러한 확장 방식은 쉬워보이지는 않을 것이다. 이 점은 노동 비용이 증가되는 시작점이므로 이에 대해서는 뒷부분에서 자세히 설명할 것이다.

비공유 데이타베이스 시스템은 "주 디스크 및 보조 디스크 소유권" 개념을 사용하여 두 개 디스크 집합이 두 개 노드와 물리적으로 연결되도록 이중 포트 디스크 하위 시스템을 사용할 수 있다. 그러한 구성은 노드 실패로 인한 시스템 비가용성에 대해 보호된다. 시스템의 비가용 상태가 표시되기 위해서는 두 개의 노드 장애가 있어야 한다. 발생 가능성이 매우 적을 지라도 단일 노드 장애로 인해 성능이 크게 저하될 수 있는데, 그 이유는 다른 노드에서 처리 중인 작업의 거의 두 배를 한 개 노드(특정 n 노드 클러스터에서)에서 처리하고 그 노드가 트랜잭션 완료에 있어 결정적인 경로 상에 있기 때문이다.

표면적인 수준에서 순수 비공유 시스템은 분산 데이타베이스와 비슷하다. 해당 노드에서 실행 중인 트랜잭션은 액세스되는 데이타를 소유하는 다른 노드에 메시지를 전송해야 한다. 또한 필요한 읽기/쓰기 작업을 수행하도록 다른 노드 상에서 수행되는 작업을 조정해야 한다(아래 2단계 커밋 참조). 이러한 메시징을 흔히 "기능 쉬핑(function shipping)"이라고 한다. 그러나 하나의 데이타 딕셔너리를 사용하여 하나의 물리적 데이타베이스를 처리한다는 점에서 비공유 데이타베이스는 분산 데이타베이스와는 근본적으로 다르다.

기존 공유 디스크

순수 공유 디스크 데이타베이스 아키텍처에서 데이타베이스 파일은 각 인스턴스가 모든 데이타에 대한 액세스 권한을 가진 느슨하게 결합된(loosely-coupled) 시스템의 노드들 간에 논리적으로 공유된다.

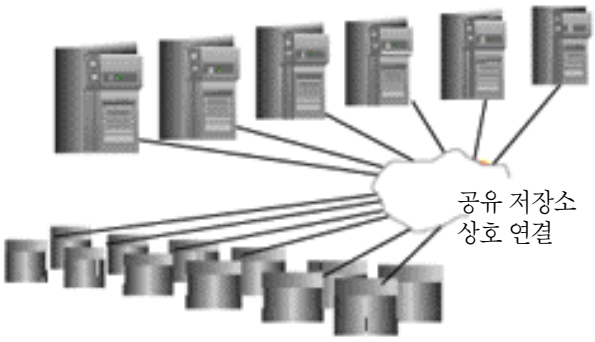

그림 3: 공유 데이타베이스 클러스터

공유 디스크 액세스는 직접 하드웨어 연결을 통해 수행되거나 모든 노드 상에 있는 모든 장치의 단일 뷰를 제공하는 운영 체제 추상화 레이어를 사용하여 수행된다. "순수 공유 디스크"는 모든 노드에서 모든 디스크에 대한 동등하고 직접적인 액세스를 사용할 수 있는 클러스터된 시스템에서 효과적인 방법으로, 공유 디스크 기법의 단일 노드 방식은 기본 SMP 시스템에 매핑된다.

이 세분화 방식은 나중에 "노드 추가"를 설명할 때 언급할 것이다.

공유 디스크 방식에서는 모든 인스턴스 상에서 실행되는 트랜잭션이 데이타베이스의 부분을 직접 읽거나 수정할 수 있다. 그러한 시스템은 상호 연결 통신을 사용하여 다수의 노드에서 수행되는 업데이트 작업을 동기화해야 한다. 두 개 이상의 노드가 동일한 데이타 블록을 차지하려고 분쟁을 벌이는 경우, 기존의 공유 디스크 데이타베이스 시스템은 다른 노드가 동일한 데이타 블록에 액세스하기 전에 데이타 상에 잠금을 설정한 노드가 디스크에 블록을 쓰는 등 다수의 노드 상에서 데이타 액세스를 동기화할 때 디스크 I/O를 사용한다. 동기화에 소요되는 디스크 I/O 비용은 기존의 공유 디스크 데이타베이스 시스템 상에서 비분할 작업 로드 또는 높은 분쟁 작업 로드를 확장하기 위한 중요한 문제들을 노출시킨다.

공유 디스크 데이타베이스의 다중 노드 확장성 특성은 읽기 위주의 애플리케이션이나 Oracle8*i* Parallel Server의 이전 버전에서 보았던 항목들을 업데이트하기 위해 데이타를 분할할 수 있는 애플리케이션에 적합하다. 나중에 설명하겠지만 일관된 캐시를 유지하는 데 드는 비용이 증가하므로 분할 작업이 올바르게 수행되지는 않는다.

공유 디스크 방법의 이점은 생존 노드가 오직 하나일 경우라도 모든 데이타에 액세스할 수 있는 상태로 유지하는 결함 허용력의 높은 수준을 제공한다는 점이다. 공유 디스크 클러스터의 노드가 실패할 경우, 시스템은 모든 생존 클러스터 노드 간에 작업 로드를 동적으로 재분배한다. 이렇게 하면 중단되지 않은 서비스와 안정된 클러스터 수준의 리소스 이용을 유지할 수 있다.

DB2 EEE 비공유

DB2 EEE가 비공유로 간주될지라도 데이타의 높은 가용성을 제공하기 위해서는 데이타베이스가 공유 디스크 상에서 생성되어야 한다. 비공유는 물리적 연결이 아닌 실행 시에 데이타의 소유권을 가리킬 뿐이다. DB2 EEE에서는 디스크 분할 시 데이타의 보조 소유자 역할을 하는 시스템의 부분 집합으로만 연결하는 것이 가능하다. 그러나 부분 집합만 사용되는 경우 일부 노드가 다른 노드보다 더 많은 작업 로드를 실행하는 등 보다 분주하게 작업을 실행하므로 전체 시스템 처리량 및 성능이 저하될 것이다. 이와는 대조적으로 RAC는 디스크와 모든 시스템 간의 전체 연결을 필요로 하므로 이러한 문제를 겪지 않는다. 이 사실을 통해 IBM의 고급 Parallel Sysplex 솔루션도 공유 디스크를 사용하여 이러한 문제를 해결함을 알 수 있다.

RAC 및 IBM의 Parallel Sysplex 공유 데이타

비용이 많이 드는 수동 조정을 많이 다루어야 한다는 점 외에도, 분할된 방법은 용량 계획, 성능 조정 및 가용성 면에서 상당한 어려움을 야기시킨다.[2]

RAC와 IBM의 Parallel Sysplex 모두는 모든 서버가 모든 데이타에 대한 액세스 권한을 가지는 "공유 데이타"("비공유"에 반대) 방법을 사용하므로 모든 트랜잭션이 모든 서버 상에서 실행될 수 있다. 작업 로드 밸런싱 방법을 사용하면 작업을 고르게 분배할 수 있으며, 이 경우 시스템 관리 지점은 하나이다. 사용자는 서버가 오프라인 상태일지라도 계속해서 애플리케이션을 사용할 수 있으므로, 거의 직선 형태의 확장성을 갖는 단일 애플리케이션에 비해 모든 서버의 결합된 능력을 사용할 수 있다. 이 점에 있어서는 IBM 제품도 마찬가지이다.

> "데이타 공유 및 작업 로드 밸런싱을 사용하면 작업을 사용할 수 있는 프로세서에 즉시 지정할 수 있으며 서버가 비용 손실을 유발하는 다운타임이 없이 클러스터에 동적으로 추가될 수도 있다. 이 작업은 다수의 서버 상에서 애플리케이션 또는 데이타베이스를 분할(이 작업은 흔히 직원이나 시스템 관리 면에서 많은 양의 새로운 투를 필요로 하는 시간과 비용이 많이 드는 프로세스임)하지 않고도 수행될 수 있다"[3]

그러므로 IBM이 이 제품을 최고의 솔루션으로 홍보하는 모든 이점들은 Oracle과 동일한 공유 디스크 아키텍처에서 찾아볼 수 있다. 또한 Oracle을 사용할 경우 구입할 수 있는 가격대의 여러 공급업체와 여러 운영체제(UNIX, Windows, Linux)가 제공된다.

하지만 Parallel Sysplex 클러스터링 기술을 완전히 활용하려면 특별한 독점 하드웨어 및 소프트웨어 요소가 필요하다4.

- 두 개의 결합된 LPAR을 최소로 설치해야 한다.
- 각 Parallel Sysplex 구성에서 두 개 이상의 결합 기능(Coupling Facilities) 및 결합(Coupling) 링크를 사용해야 한다.
- 다수의 프로세서가 Sysplex Timer에 연결되어야 한다. Sysplex Timer는 시간대(TOD)를 설정하고 Parallel Sysplex 클러스터에서 시간 동기화를 유지한다.

---

2 IBM, " Parallel Sysplex Cluster Technology: The IBM Advantage", GF22-5015-07 October 2001
3 IBM, "IBM Parallel Sysplex clustering technology", G221-4101-05 2000년 10월
4 IBM, "IBM Parallel Sysplex clustering technology", G221-4101-05 2000년 10월

그러므로 IBM이 이 제품을 최고의 솔루션으로 홍보하도록 하는 모든 이점들은 Oracle과 동일한 공유 디스크 아키텍처에서 찾아볼 수 있다. 또한, Oracle을 사용할 경우 구입할 수 있는 가격대의 여러 공급업체와 여러 운영체제(UNIX, Windows, Linux)가 제공된다.

RAC 공유 캐시

RAC는 Cache FusionTM을 사용한다. 이 기술은 고속 상호 연결을 이용하여 캐시 일관성을 유지하는 공유 캐시 일관성 기술이다. RAC는 디스크 I/O를 통한 노드 간 동기화의 요건을 제거하므로 기존 공유 디스크 클러스터로 인한 병목 현상이 해소된다.

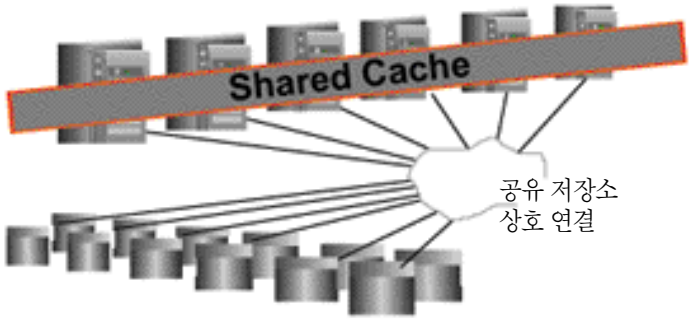

그림 4: Cache Fusion은 캐시 일관성을 위한 확장성 있는 공유 캐시를 사용하며 디스크 I/O를 제거한다.

Cache Fusion은 로컬/원격 서버 캐시 모두를 사용함으로써 모든 노드의 캐시를 활용하여 데이타베이스 트랜잭션을 제공한다. 이 점은 보다 느린 디스크 I/O의 속도를 크게 향상시켜 기존 공유 디스크에서 드러났던 이러한 기본적인 약점을 해결해 준다.

GCS(Global Cache Service)는 인스턴스의 버퍼 캐시 상에서 데이타 블록의 상태 및 전송을 관리하며 버퍼 캐시 관리자와 완벽하게 통합되어 GRD(Global Resource Directory)에서 리소스 정보를 신속히 조회할 수 있다. 이 디렉토리는 모든 인스턴스 상에 배포되며 전체적인 조정을 필요로 하는 모든 데이타 블록을 비롯한 리소스에 대한 상태 정보를 유지한다. 로컬 노드는 임의의 클러스터 데이타베이스 노드 캐시에서 필요한 블록을 직접 얻을 수 있으므로 업데이트 작업에서는 동기화를 위한 디스크 I/O가 필요 없다. 이러한 구현은 데이타베이스 캐시 작업 집합을 효과적으로 확장하며 디스크 I/O를 감소시켜 데이타베이스 작업 속도를 크게 향상시켜준다.

시스템 소프트웨어 요구 사항

RAC 및 DB2 EEE는 클러스터 관리자에 의존하여 클러스터 멤버십 서비스를 제공한다. 일부 플랫폼(Windows 및 Linux) 상에서 Oracle은 클러스터 관리자 소프트웨어를 제공한다. 그러므로 이러한 플랫폼 상에서는 업체의 클러스터 소프트웨어의 어떠한 제한 사항에도 제약을 받지 않는다. 예를 들면, Windows 2000에서 DB2 EEE는 단일 클러스터에 4개 또는 8개 노드(Windows 2000 에디션에 따라)로 제한되므로 데이타베이스 파일에 대해 MSCS(Microsoft Cluster Service)에서 원시 장치를 사용할 수 없다. 그러나, 단일 DB2 EEE는 Windows 2000을 사용하는 경우에도 8개 이상의 시스템에서 실행될 수 있다. 단, 서로 다른 그룹 내 시스템 상에서 분할에 대해 장애를 복구할 수 없다는 제한이 있다.

한편, RAC은 현재 64개 이상의 노드에서 실행될 수 있다. 이러한 장점은 보다 저렴하고 작은 노드를 사용하여 적지만 비싼 대형 노드와 동일한 작업을 수행할 수 있도록 해주며 보다 크고 강력한 클러스터로 확대하기 위해 최고 한도를 확장한다.
또한 CFS(Cluster File Systems) 및 클러스터 별칭(Cluster Alias)에 대한 지원을 통해 전체 클러스터를 하나의 대형 SMP처럼 관리할 수 있다.

DB2 EEE를 사용하는 RAC에 대한 플랫폼 요구 사항을 요약해 보면 두 개 데이타베이스에 대한 하드웨어 및 소프트웨어 요건은 거의 동일하다. 그러나, 각 플랫폼 상에서의 실제 애플리케이션 배치에 소요되는 비용을 비교하면 차이가 있을 것이다.

배치

지금까지는 하드웨어, 시스템 소프트웨어 구성 및 설치가 비슷하다는 사실을 알았다. 이제는 단일 시스템에 배치할 때와 비교하여 클러스터 상에 데이타베이스 및 애플리케이션을 배치하는 데 필요한 추가 절차를 중점으로 살펴볼 것이다.

RAC 및 DB2 EEE를 사용하면 사용자는 클러스터의 모든 시스템에 연결할 수 있다. 경우에 따라, 위치 감각을 향상시키기 위해서는 특정 시스템으로의 클라이언트 연결을 라우팅하는 것이 바람직할 것이다.(이에 관해서는 성능 부분에서 설명함) 하지만 이것만으로는 연결 구성과 관련하여 중요한 구조적 차이가 드러나지 않는다.

DB2 EE 데이타베이스는 EEE에서 사용되도록 이전되어야 한다. DB2 EEE에서 외부 하드웨어 리소스를 사용하기 위해 각 노드(분할)에서 소유하는 디스크들 간에 고르게 데이타를 분할해야 하므로 데이타는 물리적으로 언로드 및 재로드되어야 한다. 이러한 방식으로 데이타를 분할하는 작업은 여기서 설명할 매우 복잡한 작업 중 하나이다.

배타적 인스턴스에서 RAC 상에 데이타베이스를 배치하려면 DBA가 클러스터에서 예상했던 노드 수만큼 많은 재실행 스레드 및 실행 취소 테이블 공간을 생성하기만 하면 된다. 데이타베이스 파일을 수정할 필요는 없다.

대조적으로, DB2 EE 데이타베이스는 EEE에서 사용되도록 이전되어야 한다. DB2 EEE에서 외부 하드웨어 리소스를 사용하기 위해 각 노드(분할)에서 소유하는 디스크들 간에 고르게 데이타를 분할해야 하므로 데이타는 물리적으로 언로드(Unload) 및 재로드(Reload)되어야 한다. 이러한 방식으로 데이타를 분할하는 작업은 여기서 설명할 매우 복잡한 작업 중 하나이다.

데이타 분할

비공유 환경에서 데이타 분할은 일부 규칙 모음에 따라 다수의 서버 상에 데이타를 분할하는 방법이다. 보통 애플리케이션 성능을 이상적으로 향상시키기 위해 분배를 고르게 수행하고자 하는데, 그러기 위해서는 패턴 상에서 애플리케이션 및 데이타 액세스를 잘 이해해야 한다.

DB2 EEE 분할

먼저 DB2 EEE 분할을 간단히 살펴보자. 노드/분할은 nodegroup에 할당되고, 각 테이블 공간도 *nodegroup*에 할당된다. 테이블이 생성되면 그 테이블에는 하나 이상의 열로 구성된 분할 키가 할당된다. 행이 테이블로 삽입될 때 행의 분할 값은 전류 분할 맵에 기초하여 특정 분할에 할당되는 해시 테이블 안에 해시된다. 기본 분할 맵은 nodegroup의 분할 상에서 고르게 데이타를 분배하지만 뒷부분에서 성능에 영향을 미치는 치우침 또는 핫스폿을 막기 위해 DBA의 조정이 필요할 수도 있음을 알게 될 것이다.

복잡한 애플리케이션에는 몇 천 개의 테이블이 포함될 수 있다. 이 때 테이블 중 약간의 퍼센트만 주로 액세스되므로 모든 테이블이 다 최적으로 분할될 필요가 없을 수도 있다. DB2는 분할 키 사양을 갖지 않는 테이블에 대해 기본 키 또는 첫 번째 짧은 열에 기초하여 테이블을 자동으로 분할한다. 그러나 몇 개의 중요 테이블에 대한 분할 기준을 선택하려면 애플리케이션을 잘 이해해야 하는데, 이러한 이유에서 최적 분할을 위한 몇 가지 사항을 고려해야 한다.

분할 키 선택 고려 사항

고른 데이타 분배

먼저, 분할 키는 모든 논리적 노드에 데이타를 고르게 분배해야 한다. 그러기 위해서는 테이블 정의 및 열 사용을 이해해야 하는데, 이러한 점들을 파악하고 있어도 데이타를 고르게 분배할 수 없는 경우도 있다. 이러한 일은 열에서 기수(cardinality)가 낮거나 기수가 높은 열에서조차 특정 키에 대한 액세스가 균형이 잡히지 않아 불안정할 경우에 일어날 수 있다. 예를 들어, 제품 테이블이 제품 ID로 분할되는 경우 액세스는 널리 사용되는 특정 제품 쪽으로 치우칠 수 있다. DB2는 데이타가 고르게 분배되지 않을 경우 분할 맵을 변경할 자동 로더(autoloader)라고 하는 유틸리티를 제공한다. 그러나 이 유틸리티는 다음과 같은 이유에서 실제로는 유용하지 못하다.

- 분할 맵은 현재의 데이타에 기초하여 변경된다. 추후 액세스 패턴은 다양한 치우침을 가져올 수 있다.
- 분할 맵은 동일한 노드 그룹 안에 있는 모든 테이블 공간(및 이러한 테이블 공간 내 모든 테이블)에 대해 변경될 것이다.

- 데이타가 고르게 분배될 수 있지만 액세스가 한 쪽으로 치우칠 수 있다(제품 수가 많을 경우 특정 유명 제품을 고려함). 데이타 재분배를 통해 액세스 치우침을 없앨 수는 없다.
- 데이타를 재분배하면 애플리케이션 업데이트가 차단된다. 이 작업은 높은 수준의 유지 보수 오프라인 작업이다.

RAC는 노드 간 캐시 충돌을 해결하기 위해 메시지 트래픽을 투명하게 최적화한다.

테이블 콜로케이션(collocation)

또 하나 고려할 사항은 열을 조인하려는 경우에 함께 배열된 조인(예를 들면 분할 간 통신 없이 동일 노드 상에서 수행될 수 있는 조인)에 대해 허용할 분할 키를 선택할 때 각별히 주의해야 한다는 점이다. 콜로케이션(collocation)은 DB2 상에서 성능에 있어 중요한 요소이다. IBM에 따르면 콜로케이션을 구현하기 위해 설계자는 "테이블이 동일한 nodegroup 안에 있고 동일한 수의 분할 키 열을 사용하며 분할 키 열의 데이타 형식이 쌍 단위로 호환될 수 있는 경우, 다양한 테이블에서 행의 동일한 분할 키 값은 동일한 데이타 분할에 할당된다."라고[5] 설명해야 한다.

어떤 스키마에서는 모든 조인을 함께 둘 수 없다. 예를 들어, TRC-D 스키마에서 ORDER 테이블 및 ITEMS 테이블은 ORDER 키를 사용하여 함께 둘 수 있지만 PARTS 테이블과 ITEMS 테이블의 조인은 함께 둘 수 없다. 이는 PARTS 테이블이 PART NUMBER에 의해 분할되기 때문이다(PARTS 테이블에는 ORDER 키가 없음). 고속 상호 연결을 덜 중요시 할 경우 메시지 트래픽을 줄이는 일은 클러스터 데이타베이스 성능에 여전히 중요한 문제이다. RAC는 노드 간 캐시 충돌을 해결하기 위해 메시지 트래픽을 투명하게 최적화한다.

데이타 액세스 패턴

"저희 애플리케이션은 일주일 마다 변경됩니다." Amazon.com의 데이타베이스 시스템 & 엔지니어링 담당 이사 Matt Swann.

분할 키를 결정하는 데 있어 세 번째로 고려해야 할 사항은 모든 질의가 사용될 주기와 테이블 크기이다. 테이블에 적절한 공동 위치 열이 없을 경우 각 노드 상에서 복제된 사본을 만들지 못할 수도 있지만, 다음에 제시될 몇 가지 성능 상의 영향을 가져올 수는 있다.

실제 애플리케이션의 데이타 분할 문제는 비용이 많이 들고 복잡할 수 있다. 거의 잘 변경되지 않고 이러한 분석을 단 한 번만 수행할 수 있는 애플리케이션이 있는 반면, 자주 변경되는 웹 기반 소매업체와 같은 다른 많은 애플리케이션이 있다. 성공한 온라인 소매업체 중 하나인 Amazon.com은 책에서부터 여행 서비스에 이르기까지 다양한 범주에 속하는 수 백만 개의 고유한 새로운 항목 및 사용된 항목의 목록을 표시한다. Amazon.com의 데이타베이스 시스템 및 엔지니어링 담당 이사인 Matt Swann은 "저희 애플리케이션에서 사용하는 SQL 코드는 매주 바뀝니다. 이러한 변화의 페이스는 업무가

---

5 Gene Kligerman, IBM, "DB2 UDB EEE as an OLTP Database",U29 DB2 and Business Intelligence Conference, Las Vegas, 2000년 10월

요구하는 기능에 의해 강요되며 시간 소비가 많은 분석 또는 데이타 재분배로 인해 방해를 받을 수 없습니다."라고 언급했다.

기본 키 또는 고유 키

마지막으로 고유 인덱스 또는 기본 키는 분할 키의 상위 집합이어야 한다. 이 제한은 DB2 EEE가 테이블과 동등하게 분할되는 인덱스만을 지원하기 때문에 존재하는 것이다. 그러므로 애플리케이션 변화 없이 분할 열 이외의 열에 대한 고유 제약 조건을 강요해야 하는 애플리케이션을 배치할 수는 없다.

이것을 요약하면, 비공유 데이타베이스에서는 일반적으로 최적의 성능을 위해 수동으로 데이타를 분할해야 한다는 것이다. 공유 캐시 데이타베이스 클러스터는 데이타 분할 방법을 확장할 필요가 없다. 비공유 클러스터를 사용할 경우 바람직한 분할 방법을 선택하기란 어렵고 모든 노드 상에서 작업 로드의 균형을 유지하며 비용이 많이 드는 재분할 작업을 피할 수 있도록 주의하여 선택해야 한다.

공유된 캐시 데이타베이스 클러스터에서는 데이타 분할 방법을 확장할 필요가 없다.

이러한 재분할 노력은 시간이 많이 들며, 노동 집약적이고 사람의 실수(이는 다운타임의 원인 중 하나이기도 함)가 발생할 수 있다. 또한 클러스터된 시스템이 노드를 추가하여 확장될 때마다 재분할되어야 하며(노드 추가 참조), 실제로 데이타 읽기/쓰기를 포함하는 데이타베이스에 이 기술을 적용하는 일은 그렇게 간단하지는 않다.

결국, RAC 상에서는 애플리케이션을 "즉시" 분배할 수 있지만 DB2 EEE에서는 그러한 작업을 수행할 수 없게 된다.

확장성과 성능

확장성과 성능에는 여러 가지 다양한 측면이 있으며 이는 애플리케이션 유형(OLTP, DSS, 패키지 애플리케이션)과 그러한 애플리케이션이 실행되는 작업 로드에 크게 좌우된다. 여기서는 이러한 각각의 애플리케이션 유형의 몇 가지 특성들과 두 가지 데이타베이스 구조가 작업 로드의 성능에 어떻게 영향을 미치는지 살펴볼 것이다.

OLTP 애플리케이션에 대한 성능

OLTP 환경에서 일반적인 트랜잭션은 DSS(의사 결정 지원 시스템)와 비교할 때 상당히 짧다. 성능은 처리량(예를 들면, 1시간 단위에 처리되는 트랜잭션의 수) 또는 응답 시간(예를 들면 해당 트랜잭션이 수행되는 데 소요되는 시간의 양)에 따라 측정된다. 이미 앞에서 데이타 분할로 인해 애플리케이션 배치에 어떻게 복잡성이 추가되는지 살펴보았으므로, 이제는 데이타 분할이 애플리케이션의 성능에 어떻게 영향을 미치는지 살펴보자.

분할 조사

DB2 EEE는 테이블과 동등하게 분할되도록 인덱스를 제한한다. 단 하나의 분할 키만을 선택해야 하므로 해당 분할 키 이외의 키에 대한 모든 질의는 클러스터 내 모든 분할로 브로드캐스트된다. OLTP 애플리케이션에서의 중요한 테이블에 최대 10배의 인덱스를 만들려는 경우에는 보통 이렇게 하는 것이 일반적이다. 몇 개의 서로 다른 SQL where 절에 기초하여 데이타를 신속히 조회할 수 있도록 지원하는 데에는 대량의 인덱스들이 필요하다. DB2 EEE에서는 인덱스를 테이블과 동등하게 분할되도록 해야 하므로 분할 키 이외의 인덱스(또는 분할 키의 접두어)에 대한 질의로 인해 모든 분할로 브로드캐스트된다. 따라서 분할 감소(partition pruning)를 초래하지 않는 질의에 대해 다수의 분할 조사(partition probe)가 수행되는 것이다.

그림 5 :DB2 DEE 분할 조사

예를 들면, 그림 5에서 직원 테이블이 직원 수에 따라 분할되고 직원 이름에 대해 인덱스가 있는 경우 이름별로 직원을 조회하려면 DB2 EEE에서 모든 분할에 대해 직원 이름 인덱스를 조사해야 한다. 이름별로 직원을 조회하기 위해 수행되는 모든 작업은 분할 수가 많을수록 증가된다. 또한 직원 테이블에 두 개 이상의 인덱스, 예를 들면 10개 인덱스가 있을 경우, 질의의 90%가 모든 분할로 브로드캐스트된다(모든 키에 대한 질의가 고르게 분배된다고 가정할 때). 낭비된 작업을 계산하기 위해 인덱스 조회에서 x번의 cpu 주기와 y번의 논리적 IO를 소비한다고 가정하자. 클러스터에 노드 수가 n개인 경우 동일한 인덱스 조회에는 n * x cpu 주기 및 n * y 논리적 IO가 소요될 것이다. 따라서 노드를 더 추가한다 하더라도 성능이 향상되지 않는다.

하지만 이와는 대조적으로 RAC에서는 단일 B-Tree 직원 이름 인덱스에서 해당 인덱스 페이지만을 액세스하여 동일한 질의를 실행할 수 있다

데이타 캐싱

RAC에서는 GCS(전역 캐시 서비스)를 사용하여 캐시 일관성을 유지한다. 전역 캐시 서비스를 사용하면 RAC는 데이타를 필요로 하는 많은 수의 노드에서처럼 매우 드물게 수정되는 데이타를 캐시하고 캐시 안에 공간을 마련할 수 있다. 이러한 데이타에 대한 추후 액세스는 주 메모리 속도에서 수행될 수 있다. 한편, DB2 EEE 시스템은 노드 간 통신을 사용하여 마지막 액세스 이후에 수정되지 않았을지라도 다른 분할에서 데이타에 액세스해야 한다.

IBM은 노드 간 통신을 줄이려면 각 분할에 대해 테이블의 복제된 사본을 생성하라고 제안한다. IBM에 따르면, 일부 테이블이 동일한 키에 대해 분할될 수 없지만 크기 면에서 그다지 크지 않고 대체로 읽기 전용일 경우 복제 테이블(Replicated Tables)[6]을 사용할 수 있다고 한다. 복제 테이블을 사용하면 전체 테이블의 내용(또는 일부 내용)을 nodegroup 내 모든 데이타베이스 분할로 복사할 수 있다. 그러나, 복제된 테이블은 저장 공간을 낭비한다. 또한 이러한 테이블을 선택하는 작업을 자동으로 수행할 수 없고 복제된 테이블의 업데이트 작업이 성능에 영향을 주므로 복제되는 테이블을 선택할 때 각별히 주의해야 한다.

2단계 커밋

DB2 EEE 시스템 상에 있는 두 개 이상의 분할에서 데이타를 수정하는 모든 트랜잭션은 2단계 커밋 프로토콜을 사용하여 다수의 시스템 상에서 트랜잭션의 무결성을 보장해야 한다. 다시 말하면 트랜잭션을 커밋하기 전에 최소한 두 개의 로그 강요(log force)와 최소한 하나의 왕복(round-trip) 메시지를 2단계 커밋 프로세스의 일부로서 기다려야 하는데, 이렇게 하면 OLTP 애플리케이션의 응답 시간이 증가된다.

RAC에서는 커밋이 트랜잭션을 실행 중인 노드 상의 로그 강요(log force)만을 기다리면 된다. 해당 트랜잭션에서 클러스터 내 다른 노드에서 수정한 데이타에 액세스한 경우 이러한 블록은 디스크 I/O를 발생시키지 않고 고속 상호 연결을 사용하여 전송된다. RAC에서는 전송 전에 수정 사항의 로그 강요가 블록 내 존재해야 한다. 하지만 SAP 판매 및 분배 벤치마크(SAP Sales and Distribution Benchmark)와 같은 삽입 집중 벤치마크(insert intensive benchmark) 상에서도 이러한 트랜잭션의 아주 작은 퍼센트(5% 이하)만 로그 강요에 의해 차단된다. 그 이유는 다른 노드에서 필요로 하기 전에 후면에서 로그 기록자가 블록에 대한 수정 사항 로그를 디스크에 작성하도록 끊임없이 강요하기 때문이다.

한편, DB2 EEE 트랜잭션은 커밋의 첫 번째 단계에서 준비된 레코드를 기록하고 2단계 커밋의 두 번째 단계로 진행하기 전에 커밋이 강요될 때까지 기다려야 한다.

---

6 Gene Kligerman, IBM, "DB2 EEE in an OLTP Environment",C14 International DB2 User's Group conference, Orlando, 2001년 5

확장성이 극대화된 시점

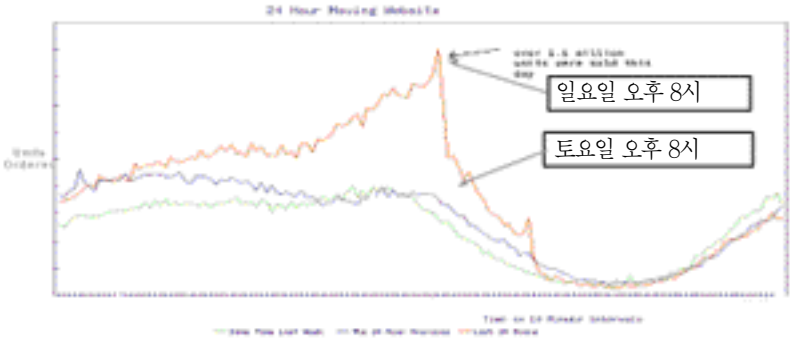

그림 6: Amazon.com 최고 로드

작업 부하 비대칭

DB2 EEE 시스템은 작업 부하 비대칭 문제를 겪을 수도 있는 두 가지 이유가 있다. 첫 번째이유는 기본 데이타가 모든 분할에 고르게 분배되지 않을 수 있기 때문이다. 특히 기수가 낮은 데이타일 경우가 이에 해당되며, 기수(cardinality)라는 용어는 집합(SET)에서 카디널(기본) 멤버의 수를 나타낸다. 기수는 유한(음수 이외의 정수)하거나 무한할 수 있다. 예를 들면, 미국의 주 집합에서 기수는 50인데 4개 노드 클러스터에서 이 집합을 분할할 경우에 2개 노드에는 12개의 주가 포함하고 다른 2개 노드에는 13개의 주가 포함한다면 어느 정도 괜찮을 것이다. 그러나 주가 지리적 영역(NE, SE, NW, SW)으로 나눠지고 인구에 따라 일부 애플리케이션을 실행했다면 NE에 해당하는 노드에서 로드가 가장 많을 것이다.

두 번째, 기본 데이타가 고르게 분배되는 경우라도 주기적인 경향 때문에 데이타 액세스가 몇 개의 열 값으로 구성된 작은 집합으로 치우칠 수 있다. RAC에서는 데이타를 소유하는 단일 노드가 없어 모든 노드에서 모든 데이타에 액세스할 수 있으므로 작업 부하 비대칭 문제를 겪지 않아도 된다. 그림 6을 살펴보면 도표에서 10분 간격마다 유닛이 주문되는데, 녹색 선은 이전 주의 주문, 청색 선은 24시간 이전 주문(토요일 오후 8시), 빨간색 선은 최근 24시간 동안의 주문(일요일 오후 8시)을 나타낸다. 빨간색 선에서 뾰족한 부분은 일일 최고 1,100,000개까지의 작업을 강요하여 신제품을 출시할 수 있게 되었기 때문이다. 그러므로 제품 ID와 같은 열의 기수가 높을 경우라도 작업은 특정 제품의 많은 요구로 인해 여전히 치우칠 수 있다.

트랜잭션 라우팅

해당되는 경우 클러스터의 노드 부분 집합으로 트랜잭션을 라우팅하여 RAC 상에서 성능을 더욱 향상시킬 수 있다.

트랜잭션 라우팅은 데이타 친화도(data affinity)를 향상시키고 노드 간 통신을 감소시킨다. 이러한 라우팅은 Oracle Net 구성의 서비스 이름을 사용하여 쉽게 수행할 수 있다. 기능에 의한 트랜잭션 라우팅은 트랜잭션에서 액세스하는 데이타의 위치를 알아야 하므로 DB2 EEE에게는 보다 부담이 되는 일이다. 또한 데이타 재분배를 사용하지 않는 보다 많은 (또는 적은) 수의 논리적 노드 상에서 트랜잭션을 실행하여 부분적으로 성능이 최적화되므로 로드 변화 면에서 덜 유연하다.

상황에 따라 RAC 시스템을 적당한 미들웨어를 사용하여 구성함으로써 애플리케이션의 바인드 값에 기초하여 요청을 라우팅할 수도 있다. 이러한 상황에서 RAC는 기본 데이타의 분할을 필요로 하지 않으며, 관리자는 트랜잭션이 라우팅되는 방법을 결정할 수 있다. 한편, DB2 EEE에서는 트랜잭션이 데이타 분할에 기초하여 라우팅되어야 하고, 해시 분할만 지원되므로(예를 들면 범위 및 목록 분할이 지원되지 않음) 라우팅 요청 시 유연성이 낮다.

DSS 애플리케이션에 대한 성능

대개 의사 결정 지원 시스템의 작업 로드가 복잡하고 오랫동안 실행되는 질의들로 구성되므로 데이타베이스 설계의 초점은 전체 경과된 질의 시간을 줄이기 위해 클러스터 내 모든 데이타베이스 노드에서 각 질의가 병렬로 실행될 수 있도록 하는 것이다.

여기서는 최적기에서 생성한 병렬 실행 계획의 품질을 비교하지 않을 것이다. 왜냐하면 이러한 비교는 클러스터에 국한되지 않기 때문이다. 같은 이유에서, 우리는 단순히 Oracle9*i*가 테이블에 대해 보다 많은 분할 옵션과 데이타베이스의 보다 나은 확장성 및 성능을 제공하는 인덱스를 지원한다는 사실만을 언급하려 한다. 또한 이미 애플리케이션 배치를 위한 다양한 데이타베이스 아키텍처의 결과와 데이타가 올바르게 분할되지 않을 경우 시스템 처리량에 대해 갖는 효과에 대해 설명했다.

다음의 두 섹션에서는 병렬 실행을 위한 IBM DB2 EEE 및 RAC 간 유사성에 역점을 두어 설명할 것이며, 그 다음으로 Oracle9*i* Real Application Cluster가 DB2 EEE 시스템보다 더 효과적으로 어떻게 클러스터에서 이용 가능한 리소스를 사용하는지를 설명할 것이다.

DSS 병렬 실행에 대한 IBM DB2 EEE와 Oracle9*i* RAC의 유사성

확장 가능한 데이타 웨어하우스에 주요 요구 사항은 데이타 볼륨이 증가할 때 하드웨어 구성 크기를 증가시켜 데이타베이스가 동일한 수준의 성능을 계속 제공할 수 있도록 다수의 CPU와 다중 노드 상에서 개별 데이타베이스 작업을 병렬화하는 기능이다.
모든 주요 데이타베이스 업체들이 공통적으로 채택하는 병렬 실행 구현을 위한 몇 가지 기술이 있다. 병렬화의 으뜸이며 가장 기본적인 구현은 분할에 기초한 병렬화 스키마다.

기본 개념은 모든 테이블을 N개의 개별 분할로 나누고 나서 이러한 분할에 대해 질의를

실행할 경우 N개의 개별 프로세스(각 분할마다 하나의 프로세스)를 사용하여 질의를 실행하는 것이다. 이 병렬화 개념에 있어 우선되는 분할 기술은 해시 분할로서, 이 분할의 경우 각 분할의 데이타 양이 동일하게 유지된다.

이 병렬 실행 아키텍처의 루트는 비공유 MPP 환경 안에 있다. MPP의 각 노드에는 단일 해시 분할이 할당되어 있다. 이 기술은 MPP의 각 노드가 데이타의 개별 분할 상에서 개별적으로 작동할 수 있도록 만들어 준다.

Oracle은 이러한 기능을 제공한다. Oracle은 해시 분할을 제공하므로 Oracle의 병렬 모델은 기능 쉬핑(function shipping) 및 분할 수준의 조인(및 그룹별 및 정렬)을 비롯한 클러스터된 환경에서 병렬 작업을 실행하는데 필요한 모든 최적화를 사용할 수 있다. 또한 직접 읽기 및 직접 쓰기를 사용하므로 DSS 애플리케이션에서 캐시 일관성 오버헤드를 피할 수 있다.

병렬 실행의 두 번째 단계는 분할 내 병렬화를 지원하는 것이다. 분할 내 병렬화를 사용하면 각 질의에 대한 병렬화 정도가 분할 수에 더 이상 제한을 받지 않는다. 대신에 데이타베이스는 각 분할에 다수의 병렬 프로세스를 제공할 수 있다.

분할 모델당 기본 한 개의 프로세스이면 노드당 단 하나의 CPU가 장착된 MPP 구성에 충분하지만, 오늘날의 병렬 하드웨어 환경에서 효과적으로 실행하려면 분할 내 병렬화가 기능의 일부로서 반드시 필요하다. 이러한 환경에서는 각 노드에 대개 최소 4개(흔히 이 개수보다 더 많음)의 CPU가 포함되어 있다.

병렬 실행 기능의 처음부터 Oracle은 병렬 실행 기능을 제공해왔기 때문에(사실상 Oracle에서 분할을 도입하기 전에 병렬화를 도입했기 때문임), 항상 모든 정도의 병렬화를 사용하여 분할 방법에 상관없이 모든 질의를 병렬화할 수 있었다.

RAC가 더 나은 이유

RAC의 병렬 질의 아키텍처는 모든 질의에 대해 병렬도를 동적으로 결정할 수 있는 능력면에서 독보적이다. 병렬도가 기본적으로 기본 테이블의 분할 방법에 의해 결정되는 DB2 EEE와는 달리, Oracle에서의 질의 병렬화는 CPU의 수, 액세스할 파일의 수 및 기타 변수에 기초하여 자동으로 결정된다.

병렬도가 기본적으로 기본 테이블의 분할 방법에 의해 결정되는 DB2 EEE와는 달리, Oracle에서의 질의 병렬화는 CPU의 수, 액세스할 파일의 수 및 기타 변수에 기초하여 자동으로 결정된다.

RAC는 필요한 병렬도를 결정하는 데 있어 적응성 있는 알고리즘을 구현하며, 이 고급 알고리즘은 병렬도를 선택할 때 데이타 웨어하우스에 대한 현재의 로드를 고려한다. 이 기능은 동시 병렬 질의를 실행하는 대형 데이타 웨어하우스의 처리량을 향상시킨다.

예를 들어, 다른 사용자가 연결되지 않은 경우 한 명의 사용자가 데이타 웨어하우스에 대해 질의를 실행한다고 가정하자. Oracle은 40의 병렬도를 사용하여 해당 질의를 실행할 수 있다. 9명의 추가 사용자가 데이타 웨어하우스에 로그인하고 질의를 제출하는

경우 Oracle은 병렬도로 4를 선택할 수도 있다. 그 이상의 사용자는 새 질의를 실행할 때 (또는 이전 사용자가 질의를 완료할 때) 새로 제출된 질의에 대해 시스템 로드에 기초하여 병렬도가 증가되거나 감소될 수 있다. 주어진 시간에 데이타 웨어하우스는 사용할 수 있는 모든 리소스를 사용하지만 어떠한 경우에도 병렬 프로세스가 너무 많아 데이타 웨어하우스 사용이 어려워지는 문제는 일어나지 않을 것이다.

RAC의 경우 적응성 있는 알고리즘에서 자동으로 클러스터 및 대량 (MPP) 환경을 자동으로 해결한다. 위의 시나리오에서 노드가 세 개인 클러스터라고 가정하자. 처음에 데이타 웨어하우스에서 실행 중인 질의가 두 개일 경우 Oracle은 12의 병렬도를 사용하여 12개의 병렬 프로세스를 클러스터의 모든 노드 상에 고르게 분배할 수 있다. 더 많은 수의 질의가 제출됨에 따라 병렬도는 그만큼 감소하게 된다. 동시 질의의 수가 6일 경우 각 질의에 대해 병렬도가 4가 됨은 물론 각 질의가 클러스터의 단일 노드로 지역화된다.

질의를 지역화하는 이 기능은 질의 실행 중에 노드 간 통신을 최소화므로 질의 성능을 향상시킨다(그림 7 참조). Oracle은 동적 병렬 질의 아키텍처에 견고한 다중 노드 기능을 제공할 수 있는 유일한 데이타베이스이므로 매우 강력한 이 기능은 Oracle 제품에서만 사용할 수 있다.

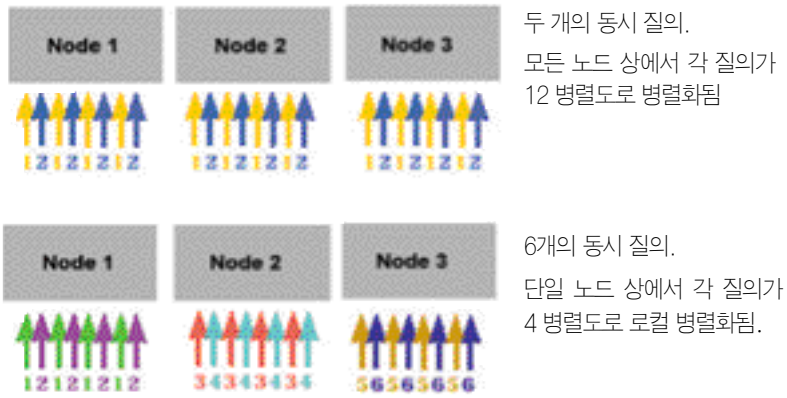

그림 7: 병렬화의 적응성 있는 알고리즘을 보여주는 예

Oracle의 실제 병렬 질의 아키텍처의 주요 이점을 요약하면 다음과 같다.

- 성능을 위해 관리성을 감수할 필요가 없다. Oracle9*i* 관리자는 가용성과 관리성 측면에서 비즈니스 요구 사항에 가장 적합한 데이타 분할 방법을 재량에 따라 선택할 수 있다. Oracle의 병렬 아키텍처는 질의 성능의 확장성 있는 사용을 투명하게 제공하지만 데이타 분할 및 다중 노드 하드웨어 아키텍처에 제한을 받지 않는다.
- 사용할 수 있는 모든 프로세스 리소스의 최대 활용. Oracle의 동적 분할 영역 내 병렬 실행을 사용하면 서버는 단일 분할이 액세스되고 있는 경우라도 다수의 프로세서 또는 노드 상에서 작업 로드를 분산시킬 수 있다.

- 모든 병렬 하드웨어 구조(SMP, NUMA, Clusters 및 MPP 시스템) 상에서의 성능, 확장성 및 관리성을 제공하는 공통의 기술 기반.

복잡한 OLTP를 사용하는 패키지 애플리케이션

Oracle, SAP, PeopleSoft 또는 Siebel의 애플리케이션 제품과 같은 유명한 OLTP 애플리케이션에는 수천 개의 테이블과 고유한 전역 인덱스가 있다. 표 1에서는 TPC-C 벤치마크와 비교했을 때의 실제 애플리케이션의 복잡성이 어느 정도인지 확인할 수 있을 것이다.

이러한 애플리케이션은 고속 데이타 액세스 및 데이타 무결성 보장, 이 두 가지 모두를 위한 기본 키 외 다른 키 열 상에서 전체 고유 인덱스를 필요로 한다. 예를 들어, Oracle eBusiness Suite에서 RA_Customers 테이블의 Customer_Number에 대해 고유 인덱스를 사용하므로 고유 비즈니스 키(기본 키 아님) 고객 번호(Customer Number)의 특정 값을 갖는 고객은 오직 한 명이다. 이러한 인덱스를 사용하지 않으면 업무용 애플리케이션 데이타가 중복될 수 있다.

	테이블	기본 키 인덱스	대체 키 인덱스	고유 SQL 문의 수
TPC-C	9	9		20
PeopleSoft	7000+	6400+	900	100,000+
Oracle eBusiness Suite (ERP 전용)	8000+	1600+	5100	290,000+
SAP	20000	16000+	2800+	273,000+

표 1: 유명 OLTP 애플리케이션의 복잡성 대 TPC-C

또한 애플리케이션은 데이타 액세스를 명확하게 분할하지 않으므로, "로컬" 데이타 액세스의 큰 부분을 차지하는 애플리케이션 테이블에 대해 분할 키를 찾을 수 없다. 로컬 액세스는 데이타의 단일 분할의 내용에서 질의의 요구 사항을 독점적으로 충족시킬 수 있는 부분이다.

Oracle eBusiness Suite, SAP 또는 PeopleSoft의 대부분의 중요 질의는 다수의 테이블을 조인하므로 다양한 질의들이 조인 조건에서 다른 대체 키를 사용한다. 또한 로컬 이외의 데이타 액세스는 자주 분산되는 트랜잭션의 허용할 수 없는 성능 오버헤드를 발생시킨다.
따라서 PeopleSoft 또는 SAP 애플리케이션을 DB2 EEE와 같은 비공유 데이타베이스 상에 배치하려면 확실한 보장이 필요한 것이다. SMP 상에서의 Oracle용으로 개발된 애플리케이션은 별도의 노력 없이도 RAC를 실행할 수 있다.

확장성

실제 OLTP 애플리케이션이 비공유 데이타베이스를 실행할 때 적절히 분배되지 못할 수 있다.

확장과 성능에 관해 논의하려면 벤치마크에 관한 모든 오정보(misinformation)를 해결해야 한다. 때로는 벤치마크 메트릭과 그 구현 방법을 이해한다고 가정할 경우, 벤치마크가 특정 작업 로드에 대해 특정 트랜잭션 집합을 실행하는 두 개 제품을 비교하기 위한 좋은 방법이 될 수 있다.

TPC-C 종합 벤치마크

TPC-C 벤치마크(현재 버전 5.0)는 OLTP 데이타베이스 및 관련 하드웨어의 확장성을 나타내는 지시 수단으로서 널리 사용된다. 이 벤치마크는 지리적으로 분산된 판매 사무실과 창고의 수가 지정되어 있는 개개 제품의 도매 공급업자용 주문 입력 애플리케이션의 일부를 모델링한다. 이 제품은 OLTP의 5가지 유형들을 포함하는 적절히 이해된 시나리오이다.

- 새 주문 – 새 고객 주문을 삽입한다. 작업 로드의 45%.
- 지불 – 계정 업데이트. 작업 로드의 43%
- 배달 – 일괄 트랜잭션. 작업 로드의 4%
- 주문 상태 – 상대 주문 상태. 작업 로드의 4%
- 재고 수준 – 웨어하우스 재고 상태를 모니터링함. 작업 로드의 4%

이 벤치마크는 지불, 주문 상태, 배달 및 재고 수준 트랜잭션의 특정 퍼센트를 강제로 결합한 상태에서 새 주문 트랜잭션/분(tpmC)을 측정한다. 그 결과는 클러스터와 비클러스터 결과라는 두 가지 범주 아래 TPC(Transaction Processing Council) 웹 사이트(http://www.tpc.org)에 보고된다. 그러므로, 이렇게 인위적으로 생성된 특정 속성을 지닌 벤치마크(실제 회사의 작업 로드가 아님)를 사용하여 상상할 수 있는 모든 하드웨어 및 소프트웨어 구성에 대한 많은 결과를 어떻게 얻을 수 있었는지 살펴볼 수 있다.

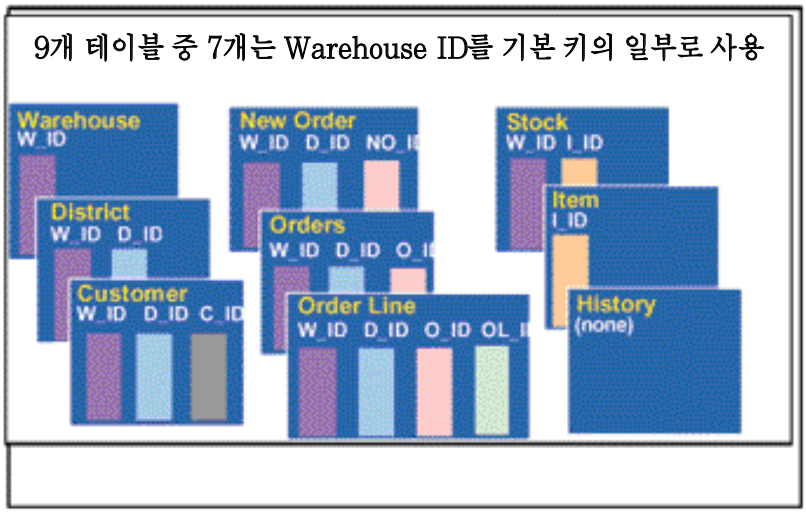

그림 8: TPC-C 스키마

TPC-C 스키마는 본래부터 분할 가능하다

TPC-C 스키마는 9개의 테이블로 구성되며, 이 중 7개는 Warehouse_ID를 기본 키의 일부로 사용한다. 트랜잭션은 기본 및 보조 키 모두를 통해 데이타를 액세스하여 이러한 9개의 테이블에 대해 작동한다.

트랜잭션의 90%는 응답 시간이 5초 이하여야 하지만, 예외적으로 재고 수준은 응답 시간이 20초 이하여야 한다. 대부분의 테이블이 웨어하우스 식별자(ID)를 참조한다는 사실 때문에 이 벤치마크는 분할된 데이타베이스에 매우 적합하다. 이 벤치마크에서는 그러한 테이블을 모을 수 있기 때문이다. ITEM 테이블만 다른 키(HISTORY에는 키가 없음)를 사용하지만 그 키는 작고 모든 분할 상에서 쉽게 복제할 수 없다.

대부분의 TPC-C SQL 액세스는 로컬이다

TPC-C 스키마는 Warehouse_ID를 분할 키로 사용하여 쉽게 분할할 수 있다. 이 분할을 사용할 경우 데이타 액세스의 대부분(99% 이상)이 로컬이다(트랜잭션을 실행하는 노드만 해당 노드의 로컬 데이타를 액세스함):

트랜잭션 유형	결합 비율(%)	"로컬" SQL 비율
새 주문	45	99.5
지불	43	95
주문 상태	4	100
배달	4	100
재고 수준	4	100

표 2: 분할된 TPC-C 스키마에서 로컬 데이타 액세스

비공유 데이타베이스에서 얻어진 TPC-C 벤치마크는 실제 OLTP 애플리케이션의 성능에 대한 타당성이 부족하다.

따라서, TPC-C 벤치마크 구현은 사실상 대부분의 로컬 트랜잭션을 실행하는 대부분의 독립적 데이타베이스의 수에 불과하다. 다양한 공급업체들이 느슨하게 결합된 구성에서 매우 많은 CPU를 사용함으로써 매우 많은 TPC-C를 얻었지만, 실제 애플리케이션에서 이러한 종류의 작업 로드를 찾기란 쉽지 않다. 데이타와 트랜잭션 중 어떤 것도 잘 정의된 분할에 거의 맞춰질 수 없다.

이제는 TPC-C 벤치마크를 실행하는 것이 하드웨어 또는 소프트웨어 성능을 테스트하는 일을 의미하는 것은 아니지만, 현재 레코드를 차단할 수 있을 만큼 충분한 하드웨어가 갖춰졌는지를 테스트하는 데 활용된다.

데이타베이스 고객인 벤치마크 데이타의 소비자는 "클러스터된" 벤치마크 결과를 허용하기 전에 TPC-C 벤치마크의 이러한 특성들을 알고 있어야 한다. 이제는 TPC-C 벤치마크를 실행하는 것이 하드웨어 또는 소프트웨어 성능을 테스트하는 일을 의미하는 것은 아니지만, 현재 레코드를 차단할 수 있을 만큼 충분한 하드웨어가 갖춰졌는지를 테스트하는 것이다.

패키지 애플리케이션 벤치마크

많은 회사들이 SAP, People Soft, Siebel 및 Oracle eBusiness Suite와 같은 패키지 애플리케이션으로 나아감에 따라 애플리케이션 벤치마크는 플랫폼 상에서의 성능을 비교하는데 사용되고 있다. Oracle 및 SAP와 같은 대규모 ISV는 여러 벤치마크를 사용하여 클러스터된 환경에서 다양한 작업 로드를 테스트한다. 이러한 벤치마크의 다양성은 고객 환경에 대한 보다 실제적인 시야를 제공하지만 모든 벤치마크의 경우처럼 결과를 해석할 때 각별히 주의해야 한다.

특수한 벤치마크로서 SAP의 Sales & Distribution Parallel Standard Application Benchmark(SD-Parallel)가 있다. 각 구성 요소의 벤치마크 트랜잭션은 일반적으로 설치(예를 들면 일일 주문수, 상품 이동 수 등)의 데이타 처리량을 반영하게 된다. 그러나, 고객이 정의한 보고서의 리소스 소비가 데이타 볼륨에 따라 좌우되므로 벤치마크 트랜잭션은 보고 작업을 반영하지 않는다. SAP Standard Application Benchmark는 주문 라인 품목이 처리되는 방법을 보여주는 전체 업무 작업 흐름을 시뮬레이트하는 몇 개의 스크립트 파일로 이루어진다.

- 주문 생성
- 이 주문에 대한 배달 정보 생성
- 주문 표시
- 배달 변경
- 상품 문제 게시
- 주문 목록 작성
- 송장 작성

SAP에는 20,000개 이상의 테이블(20,000개 이상의 인덱스)이 설치되어 있지만 SD Parallel은 사실상 빠른 동시 삽입율을 통해 벤치마크에서 140개의 신생 테이블을 사용한다. 이 벤치마크는 삽입되고 질의되는 테이블 중 여러 테이블에서 LONG RAW 열을 사용하는데, 예를 들어 삽입된 행은 4K 및 그 이상으로 길어 테이블이 빠르게 성장하도록 한다.

이를 테면, UPDATE 프로세스에서 사용하는 약 75개의 사용자 SQL 문과 DIALOG 프로세스에서 사용하는 150개의 사용자 SQL 문이 있다.

모든 변경 사항(또는 변경 요청)이 테이블 집합에 게시되므로 이러한 테이블에서 과도한 업데이트 작업을 겪게 될 수도 있다. DIALOG 프로세스는 이러한 테이블에 변경 사항을 게시하며 UPDATE 프로세스는 이러한 테이블을 처리하고 데이타베이스에 실제 변경 사항을 적용한다.
벤치마크에서는 응답 시간을 중요시하며 데이타베이스에서의 응답 시간이 일정해야 한다. DIALOG 사용자는 업데이트 요청을 한 후(또는 두 개의 다이알로그 단계 이후) 20초 안에 UPDATE 프로세스에 의해 데이타가 적용될 것이라고 예상한다. 데이타가 존재하지 않을 경우에는 벤치마크가 실패하게 된다. 마찬가지로, 평균 다이알로그 응답 시간이 2초 미만일 경우에만 벤치마크를 증명할 수 있다. 이러한 이유로 인해 업데이트 프로세스의 처리 속도와 다이알로그 프로세스의 처리 속도 간에는 그 균형이 신중하게 유지되어야 한다.

데이타 분배는 병렬 환경에서 벤치마크 결과에 크게 영향을 미칠 수 있으므로 SAP benchmark Council은 병렬 벤치마크 안에서 결과를 재현하고 비교하기 위한 수단을 만들기 위해 데이타 분배 규칙을 제정하였다. 추가 규칙은 다음과 같다.

> 사용되는 벤치마크 클라이언트에 대해 모든 데이타베이스 노드 상에 벤치마크 사용자를 고르게 분배한다(Round-Robin-Method).

다시 말해서 이 규칙은 특정 클라이언트의 사용자가 모든 노드 상에서 고르게 분배된다는 뜻이다. 이 벤치마크와 데이타/노드 유사성(분할)은 없다. 따라서 노드 간 트래픽과 캐시 간 전송의 양이 많아진다.
SAP Standard Application Benchmarks는 데이타베이스 요청 시간, 대기 시간, CPU

이용, 지정된 벤치마크 사용자들로부터의 평균 다이알로그 응답(각 다이알로그 단계마다 10초의 일정한 판단 시간 제공)과 성취된 처리량과 같은 모든 성능 관련 매개변수를 측정한다.

2001년 11월 Oracle과 Compaq은 SAP R/3 4.6C와 Oracle9*i* Database with Real Application Clusters가 초기 성능 테스트에서 훌륭한 확장성 및 처리량을 나타냈다고 발표했다. 테스트 환경에는 16개의 프로세서가 장착된 4노드 Compaq Tru64 UNIX ES4 AlphaServer 클러스터 상에서 병렬 SD 작업 로드를 실행하는 5SAP(r) R/3(r) 4.6C 및 Oracle9*i* Real Application Clusters가 제공되었다.

RAC 기술의 견고성으로 인해 배치에 있어서 요구 시 용량 계획 및 유연성을 제공할 수 있는 선형에 가까운 확장력의 기초를 제공할 수 있게 되었다. 1노드 구성에서 2노드 구성으로 바꿀 경우 1.85 인수만큼 성능이 증가되며, 1노드 구성에서 4노드 구성으로 바꿀 경우 3.24 인수만큼 증가된다.

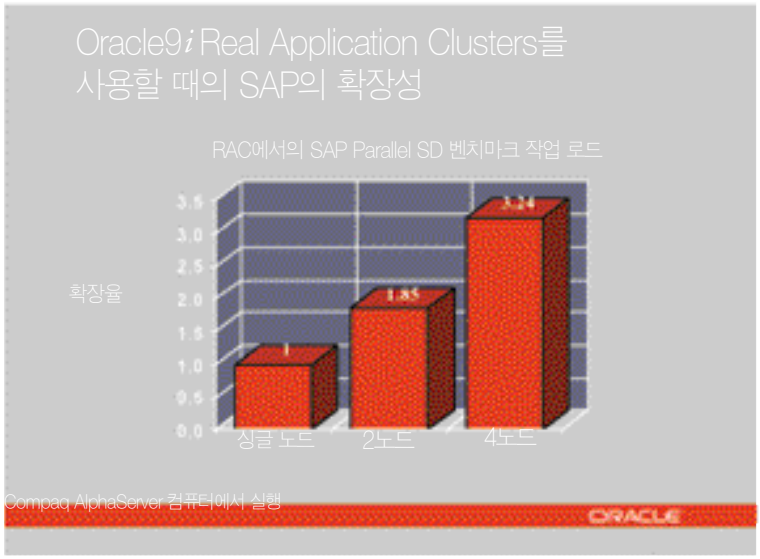

그림 9: Real Application Clusters를 사용할 때의 SAP의 확장성

이제는 TPC-C 벤치마크를 실행하는 것이 하드웨어 또는 소프트웨어 성능을 테스트하는 일을 의미하는 것은 아니지만, 현재 레코드를 차단할 수 있을 만큼 충분한 하드웨어가 갖춰졌는지를 테스트하는 데 활용된다.

SD 벤치마크는 시스템 상에 많은 로드를 부과한다는 점 때문에 이 특수한 벤치마크는 ERP(Enterprise Resource Planning) 산업에서 실제 표준이 되었다. 하드웨어 파트너는 mySAP.com 비즈니스 솔루션의 현재 사용할 수 있는 성능 및 확장성과 테스트된 플랫폼에 대한 정보를 ERP 환경의 고객 및 관계에게 제공하기 위해 벤치마크 테스트 결과를 게시한다.

패키지 애플리케이션의 시장 점유율

벤치마크가 두 개 데이타베이스를 비교하기 위한 지침을 제공하지만, 실질적인 관건은 시장의 수이다. 파트너들이 어떠한 데이타베이스를 사용하고 있는가? 경쟁업체에서는 어떠한 데이타베이스를 사용하고 있는가? IBM 측에서 선두 ISV와의 강력한 협력 관계로 DB2 판매가 증가되었다고 주장할지라도, 분석가와 ISV 자체 분석에 따르면 IBM DB2의 주식은 감소되는 반면, SAP, PeopleSoft 및 Siebel과 같은 엔터프라이즈 애플리케이션에서 데이타베이스로서 Oracle의 주요 제품은 계속 성장하고 있다고 한다.

지난 봄에 Evans Data Survey에서 600명 이상의 개발자들을 대상으로 실시한 설문조사에 따르면, Oracle Database가 애플리케이션 개발 부문에서 1위를 차지했다고 발표했다.

조사에 따르면 대부분의 주요 패키지 애플리케이션 업체에서는 Oracle 데이타베이스를 최우선으로 지원한다고 한다. 이는 ERP, CRM, 조달 및 공급 체인을 비롯한 모든 범주에 적용된다. Oracle의 시장 점유율은 1999년 49%에서 2000년 50%로 5% 증가되었지만, IBM의 수입은 14%% 하락하였으며 시장 점유율도 그 만큼 감소되었다[7].

또한 DB2 UDB보다 Oracle에 배치된 패키지 애플리케이션의 수가 더 많다. 사실상, 2001년 5월 Gartner Group 보고서에 따르면 모든 SAP 구현의 71% 이상이 Oracle 데이타베이스에서 실행되었으며, 이에 반해 DB2 UDB8에는 오직 8%만이 배치되었다. Oracle의 대규모 설치 기반에 때문에 DB2 UDB보다는 Oracle 상에 패키지 애플리케이션을 배치해 본 구현자의 수가 훨씬 더 많다. 6개월마다 SAP에서는 DB2 UDB DBA보다 128배나 더 많은 Oracle DBA를 양성하고 있다[9].

Oracle은 Unix와 Windows 상의 모든 새로운 구현 중 64%를 차지하는 Siebel의 구현에 대해 우선 데이타베이스를 유지한다[10].

마지막으로, 다른 개별적인 확증을 필요로 할 경우 Siebel, SAP 및 PeopleSoft의 웹 사이트를 참조할 수 있다. 고객 사례 연구에서 보면 60-70%가 Oracle을 사용했으며 DB2의 경우에는 5% 밖에 안되는 소수에 불과했음을 알 수 있다.

가용성

RAC 및 DB2 EEE는 고 가용성(온라인 및 오프라인 백업, 복원 및 복구)을 유지하기 위한 기존의 모든 방법들을 지원한다. 그러나, Oracle9*i* Database에서의 이러한 각각의 기능(대기 데이타베이스, 고속 재시작 복구, 블록 수준의 복구 및 플래시백 질의)에 대한 자세한 내용을 살펴보면 Oracle에서 다운타임을 최소화하여 시스템의 TCO를 감소시키기 위해 제공하는 이점들을 이해하게 될 것이다. 여기서는 단순히 하나의 클러스터 환경에 있는 두 개 데이타베이스를 비교할 것이므로 이러한 기능의 대부분은 고 가용성에 대한

---

7 AMR Research, "Field Survey of Database Best Practices", 2001년 9월
8 Gartner Group, "SAP Platform Choices: What Are the Current Trends?", 2001년 5월
9 SAP Training Schedule, 2001년 11월
10 Tom Siebel, Q2 Earnings Call, 2001년 8월

다른 설명 자료에서 설명할 것이다. 지금은 다양한 클러스터 데이타베이스 아키텍처가 데이타 가용성에 어떠한 영향을 미치는 지와 시스템 실행에 소요되는 비용을 살펴볼 것이다.

계획에 없는 다운타임

예측하지 못한 노드의 장애 시, 양 시스템 모두 데이타베이스를 투명하게 복구할 수 있다. 여기서는 단일 노드 데이타베이스에도 해당될 수 있는 이러한 알고리즘의 복구 시간을 다루지 않는다. 우리는 다음과 같은 이유에서 RAC를 사용할 때 복구 시간이 더 빠르다고 예상한다.

1. DB2 EEE는 클러스터 관리자에 의존하여 유효 노드 상에서 분할을 재시작한다. 이 작업을 수행하려면 DB2 프로세스를 시작하여 공유 메모리를 초기화한 다음 데이타베이스 파일을 열어야 한다.
2. 경우에 따라 RAC는 유효 노드의 캐시에 이미 존재하는 블록 상에서 실패한 노드가 생성한 재실행 작업을 적용할 수 있으며, 이러한 경우에는 블록을 디스크로부터 읽어올 필요가 없다.
3. 가장 중요한 이점은 애플리케이션에서 필요로 하는 데이타 및 패키지가 유효 노드에 이미 캐시되었을 수도 있으므로 데이타베이스가 복구된 후에 애플리케이션이 원래의 응답 시간보다 더 빠른 응답 시간을 얻게 될 수 있다는 점이다. 또한, 캐시의 크기가 점점 증가되고 있는 추세이므로 DB2 EEE는 장애 복구된 분할에서 캐시를 워밍업하는 데 보다 더 오랜 시간이 소요된다.

RAC에서는 실패한 연결이 유효 노드 상에 고르게 분배될 수 있다. DB2 EEE에서는 기본 클러스터 관리자가 실패한 분할에 속하는 디스크의 작업을 어떠한 유효 노드가 인계할 것인지를 결정한다. 작업 부하 비대칭을 피하기 위해 DB2 EEE는 각 유효 노드가 동일한 양의 데이타에 대한 소유권을 인계하는 등의 형태로 구성되어야 한다. 그러기 위해서는 각 노드 상에 다수의 분할을 생성하면 된다. 예를 들어, 네 개의 노드가 있을 경우 총 12개 분할이 되도록 세 개 분할이 각 노드 상에 생성된다[11]. n개 노드가 있을 경우 모든 실패 시나리오에서 유효 노드에 분할을 고르게 분배하려는 경우 분할의 수는 n, n-1, ...,1의 최소 공배수와 같다. 각 분할에 대해 노드 상에 분할이 고르게 분배되도록 클러스터 소프트웨어(HACMP, MSCS 등)를 사용하여 우선 소유자 또는 인계 목록이 생성된다.

명확히 말해서 고 가용성 구성에서는 노드 수가 증가함에 따라 분할 수도 빠르게 증가하는데, 이러한 점은 몇 가지 문제를 야기시킨다.

1. 클러스터를 관리하는 데 보다 많은 작업이 소요된다. 각 분할마다 고유의 구성 매개변수, 데이타베이스 파일 및 관리해야 할 재실행 로그가 있다.

2. 각 물리적 노드의 리소스가 충분히 활용되지 못할 수도 있다. 동일한 물리적 노드가 다수의 분할을 소유할 지라도 분할들이 버퍼 풀, 패키지 캐시 등에 사용하기 위해 메모리를 공유할 수 없다. 이로 인해 활용 부족의 문제가 발생되는데, 이는 단편화된 메모리 조각보다는 메모리의 단일 부분을 사용하는 편이 더 나을 수도 있기 때문이다.
3. 분할 수가 증가함에 따라 로드 및 데이타 치우침의 가능성도 그만큼 증가된다.
4. 분할 수가 증가함에 따라 해당 작업 로드에 대한 프로세스 간 통신도 그만큼 증가한다. 예를 들면, 네 개의 논리적 노드로 확장하는 애플리케이션이 12개의 논리적 노드로 확장될 수 있는 것은 아니다. 그러나, 고 가용성을 위해 DB2 EEE는 애플리케이션을 12개 분할 상에서 강제로 실행하도록 한다.

노드 추가

새 분할 맵에 따라 분할된 데이타를 다시 분배해야 하므로 비공유 환경에서 데이타를 분할할 경우 비용과 시간이 많이 들여 새 서버를 클러스터에 추가해야 한다. 이 때 DBA 또는 시스템 관리자는 다음과 같은 방법으로 DB2 EEE 데이타베이스에 노드를 추가해야 한다.

Oracle 9i Real Application Clusters를 사용할 경우 공유 디스크 클러스터에 완벽하게 노드를 추가할 수 있다.

- 하드웨어 추가
- 새 분할 구성(분할 특정 매개변수 등 설정)
- 보다 많은 수의 분할 상에 데이타를 분산시키기 위해 데이타를 재분배.

DB2 EEE 시스템에 데이타를 재분배할 경우 데이타베이스에 대한 DBA 작업 및 다운타임이 수반된다. 이 데이타를 재분배하는 방법에는 세 가지가 있지만 이러한 방법 모두 데이타베이스 작업을 중단시킨다.

- 기존 nodegroup 재분배
- 새 nodegroup 생성
- 부분 범위의 재분배

기존 nodegroup의 재분배

nodegroup의 데이타는 명령이 완료될 때까지 액세스할 수 없다. 재분배해야 할 데이타의 양이 많을수록 명령이 완료되는 데 소요되는 시간도 그만큼 증가된다. 이 재분배 작업은 현재 위치에서의 재분배 작업이므로 작업이 기록되고 로그 공간이 다 찰 경우 그 이후의 로그 기록이 잘려진다.

---

11 Nomani, Aslam and Pham, Lan IBM, DB2 Universal Database for Windows High Availability Supporting Using Microsoft Cluster Server – Overview, TR-74.176 2001년 5월

새 nodegroup 생성

이전 테이블의 복제본은 새 nodegroup 안에 생성된다. 이 작업을 수행하려면 이전 nodegroup에 저장된 데이타를 저장할 수 있는 충분한 공간이 필요하다. 이 공간이 있어도 새 nodegroup에 복사되고 있는 동안에는 이전 nodegroup의 데이타를 수정할 수 없다. 또한, 인덱스, 트리거, 제약 조건 및 권한과 같은 모든 종속 요소들을 다시 생성해야 한다.

낱낱으로 재분배:

이 작업은 해시 버켓에 있는 데이타를 한 번에 하나씩 재분배할 수 있다는 점을 제외하고는 첫 번째 옵션과 비슷하다. 이 옵션은 사용자가 제어하는 긴 시간 동안 재분배를 수행하므로 해당 시간에 창을 사용할 수 없도록 제한한다.

이는 엄청난 관리 부담을 만드는 것이며, 데이타베이스는 오랜 시간 동안 오프라인 상태로 남아 있게 된다.

한편, RAC에 노드를 추가할 때 필요한 관리 작업을 생각해보자.

- 하드웨어 추가
- 새 인스턴스 구성(인스턴스 특정 매개변수 등)

바로 이것 뿐이다. 데이타를 재분할하거나 오프라인 상태를 유지해야 할 필요가 없으며, 이러한 작업만으로 완벽하게 확장할 수 있다. 이렇게 RAC에서는 데이타베이스 액세스를 중단하지 않고 노드를 추가하는 것이 가능하다.

결론

이 문서에서는 RAC와 비교함으로써 DB2 EEE를 사용할 경우 배치 및 성능 면에서 많은 결점이 있음을 알 수 있었다. 또한 RAC를 다양한 애플리케이션 및 시나리오에서 사용할 수 있음을 명확히 알 수 있었다. 하드웨어 및 소프트웨어 요구 사항이 비슷할 지라도 DB2 EEE는 비공유 솔루션을 구현하는 동안 사용자에게 많은 부담을 안겨주는 것이다.

반면 RAC의 공유 캐시 아키텍처는 엔터프라이즈 eBusiness 애플리케이션 배치 시 많은 주요 장점들을 제공한다.

애플리케이션이 복잡해지거나 작업 로드가 보다 동적이 될 수 있도록 하려면 그만큼 Oracle 9i Real Application Clusters의 필요성은 더욱 커진다.

- ***모든 유형의 애플리케이션에 대해 유연하고 간편한 확장성 기능;***
  애플리케이션 사용자는 단일 고성능 클러스터 서버에 로그온할 수 있다. 데이타베이스에 쉽게 노드를 추가할 수 있으며 프로세서 노드가 추가되거나 비즈니스 요구 사항이 변경되는 경우 데이타를 분할하는 데 사용자에 의한 수동 중재가 필요없다. 또한 수정이 필요 없이 클러스터 확장성으로 인해 모든 애플리케이션을 최신 상태로 유지할 수 있다.

- ***기존 클러스터 데이타베이스 아키텍처보다 더 높은 가용성을 제공하는 솔루션***; 이 아키텍처는 고객에게 하드웨어 및 소프트웨어 구성 요소 실패에 의한 중단 시간이 최소인 거의 연속적인 데이타 액세스를 제공한다. 이 시스템은 다수의 노드 실패에 대한 회복성이 우수하고 구성 요소 실패가 최종 사용자는 인식할 수 없도록 숨겨진다.
- ***단일 관리 엔티티:.*** 단일 시스템 이미지가 모든 관리 작업을 위해 클러스터 상에서 유지된다. DBA가 설치, 구성, 백업, 업그레이드 및 모니터링 기능을 한 번에 수행하고 나면, Oracle은 관리 기능을 해당 노드에 자동으로 배포한다. 다시 말해 DBA가 하나의 가상 서버를 관리하는 것이다.
- ***총 소유 비용의 절감***; 비용이 많이 드는 많은 수동 중재 작업을 처리해야 하고 용량 계획, 성능 조정 및 가용성 면에서 큰 어려움을 야기시키는 비공유 경쟁제품과는 달리, RAC의 공유 캐시 아키텍처에서는 가장 많이 요구되는 실제 작업 로드를 처리하기 위해 거의 직선에 가까운 확장성으로 고르게 무한대로 확장될 수 있다.

한국오라클(주)
서울특별시 강남구 삼성동 144-17
삼화빌딩
대표전화 : 2194-8000
FAX : 2194-8001

한국오라클교육센타
서울특별시 영등포구 여의도동 23-10
SK증권빌딩 11층(사무실)
19 · 20층(강의실)
대표전화 : 3779-4000
FAX : 3779-4100 1

대전사무소
대전광역시 서구 둔산동 929번지
대전둔산사학연금회관 18층
대표전화 : (042)483-4131 2
FAX : (042)483-4133

대구사무소
대구광역시 동구 신천동 111번지
영남타워빌딩 9층
대표전화 : (053)741-4513 4
FAX : (053)741-4515

부산사무소
부산광역시 동구 초량동 1211 7
정암빌딩 8층
대표전화 : (051)465-9996
FAX : (051)465-9958

울산사무소
울산광역시 남구 달동 1319-15번지
정우빌딩 3층
대표전화 : (052)267-4262
FAX : (052)267-4267

광주사무소
광주광역시 서구 양동 60-37
금호생명빌딩 8층
대표전화 : (062)350-0131
FAX : (062)350-0130

고객에게 완전하고 효과적인
정보관리 솔루션을 제공하기 위하여
오라클사는 전 세계 145개국에서
제품, 기술지원, 교육 및
컨설팅 서비스를
제공하고 있습니다.

http://www.oracle.com
http://www.oracle.com/kr

제품구입문의

수신자부담 전화번호 : 00368-440-0051 수신자부담 팩스번호 : 00368-440-0062 E-Mail문의 : oracleisd_kr@oracle.com