메인 메모리 데이터베이스 시스템을 위한 이중화 설계

이 인선
신구대학 컴퓨터 정보 처리과
inseon@shingu.ac.kr

A Replication Design for Main Memory Database Systems

In-Seon Lee
Computer Information Processing Dept. of Shingu College

요 약

모든 데이터가 메인 메모리에 상주하는 메인 메모리 데이터베이스 시스템(Main Memory Database System: MMDB System)은 트랜잭션 수행중 자료 입출력을 위한 디스크 액세스를 유발하지 않으므로 전체 시스템 성능을 크게 향상시킬 수 있다. 80년대 중반에 처음으로 이 시스템이 소개된 후 현재까지 많은 연구가 이루어지고 있으며, 최근에는 램의 가격이 하락하고, 대용량화되면서 데이터의 실시간 조회, 갱신이 필요한 금융, 증권, 통신 등 여러 분야에서 MMDB 시스템의 상용화가 가시화되고 있다. MMDB 시스템의 상용화가 늘면서 더 많은 트랜잭션의 처리를 수행하고, 시스템 고장 등으로 인한 서비스 중지와 같은 결함이 일어나지 않는 시스템에 대한 요구 또한 커지고 있다. 이 요구를 만족시키는 가장 적절한 해결책으로 이중화 시스템(replication system)을 들 수 있다. 그러나, 기존의 디스크 기반 분산 데이터베이스 시스템을 위한 이중화 기법을 그대로 MMDB에 적용하게 되면 최적의 성능을 가질 수 없게 된다. 그러므로, MMDB 시스템의 특성을 잘 파악하여 이 시스템에 적합한 새로운 이중화 시스템의 개발이 필요하다. 본 논문에서는 먼저 MMDB 시스템의 구조에 대해 고찰하며, 디스크 기반 분산 데이터베이스 시스템에 적용하고 있는 이중화 기법들의 장단점을 분석한다. 이 분석을 토대로 하여 MMDB시스템에 이중화 기법을 적용하기 위해 고려해야 할 점들을 정리하고, 제시한 고려 사항들을 모두 만족하는 MMDB 이중화 시스템을 설계하였다.

1. 서론

메인 메모리 데이터베이스(Main Memory Data Base: MMDB)시스템은 데이터의 대부분 또는 모든 부분이 메인 메모리에 상주하는 데이터베이스 시스템으로 트랜잭션의 연산 수행시 디스크 입출력이 발생하지 않으므로 시스템 성능이 매우 향상되어 기존의 디스크를 기반으로 하는 데이터베이스보다 더 빠른 응답시간과 더 높은 트랜잭션 처리능력이 장점이 된다[Gar92]. 최근 들어 램의 용량이 대용량화되고 가격도 실용화할 수 있을 정도로 하락하면서 여러 상용 MMDB 시스템들이 선을 보이고 있으며[TTT99][JLR+94], MMDB시스템의 특성상 트랜잭션의 빠른 처리가 요구되는 통신, 금융, 실시간 분야 등에 적합하기 때문에 이런 산업부분에 먼저 도입되고 있는 추세이다. 이런 시스템에서 처리하는 트랜잭션의 특성을 살펴보면 길이가 짧고 대량으로 발생되며 예측 가능하고 신속한 응답시간을 요구하고 있다. 또한 적용분야의 특성상 시스템 고장으로 인해 전체 트랜잭션이 수행되지 못하는 상황이 발생해서는 안된다. 이런 트랜잭션의 특성을 고려한 해결책으로 이중화시스템(Replication System)을 들 수 있다. 이중화 분야는 디스크 기반 분산 데이터베이스 시스템에서 많은 연구가 이루어지고 있으나, MMDB시스템에 적절한 이중화시스템 개발에 대해서는 아직까지 많은 연구가 이루어지지 않고 있다. 본 논문에서는 MMDB시스템을 이중화하기 위해 먼저 MMDB의 특성을 정리하고, 이러한 특성을 반영한 이중화 시스템을 설계하였다.

본 논문의 2장에서 MMDB시스템의 구조에 대해 지금까지 이루어진 연구중 이중화시스템 설계에 필요한 내용들을 정리한다. 3장은 디스크기반 분산 데이터베이스 시스템에 적용되고 있는 이중화기법들에 대해 기술하고, 4장에서 MMDB특성을 고려한 이중화 시스템을 설계하며, 5장에서 결론을 맺는다.

2. 메인 메모리 데이터 베이스 시스템에 대한 연구

● 동시성 제어

MMDB 시스템에서 한 개 트랜잭션의 수행과정을 살펴보면 트랜잭션 연산작업에 필요한 모든 데이터가 메인 메모리에 상주하므로 데이터 때문에 디스크 입출력을 기다리는 일이 전혀 없으므로 한번 수행을 시작한 트랜잭션은 외부의 간섭이 없다면 모든 연산을 순서대로 끊임없이 실행할 수가 있게 된다. 그러므로, 동시성을 높이기 위해 여러 트랜잭션을 동시에 수행시키면서 "로킹방식"등과 같은 알고리즘에 의해 동시성 제어를 하다면, 로크 관리를 위한 메타데이터 관리비용, 트랜잭션간의 컨텍스트-스위칭 비용, 교착상태 발생에 의한 트랜잭션 철회등과 같은 오버헤드가 발생하여 결과적으로 순차수행보다 낮은 시스템 성능을 가질 수도 있다. [Inse99]에서 "두단계 로킹(2PL)방식", "타임스탬프 방식", "순차수행 방식"과 같이 세 가지 동시성 제어 방식들을 MMDB시스템에 적용했을 때의 성능을 모의실험을 통해 분석하였다. 그 결과 '순차수행방식'이 교착상태나 타임스탬프에 의한 트랜잭션 철회가 없고, 동시성 제어를 위한 오버헤드가 없다는 장점 때문에 결과적으로 우수한 시스템 성능 및 빠른 트랜잭션 응답시간을 가짐을 보였다. 또한, [BK02]에서 제안한 "almost-serial"방식은 모든 트랜잭션들이 순차적으로 수행되며, '읽기 전용' 트랜잭션의 신속한 수행을 위해 선완료(Precommit)방식을 적용하였으며, 이 경우 완료되지 않은 갱신 데이터를 읽은 트랜잭션이 먼저 완료되는 것을 방지하기 위해 "타임스탬프", "Mutex Array"를 이용하였다. 이 방식의 성능 분석을 AT&T의 상용화 시스템인 DataBlitz에서 실시한 결과, "almost-serial"방식이 "두 단계 로킹"보다 우수한 성능을 가짐을 보였다.

● 체크포인팅

체크포인팅 작업은 시스템에 고장이 났을 때 가장 최근의 DB로 복구하는 데 걸리는 시간을 단축하기 위해 평상시 주기적으로 갱신된 데이터가 있는 페이지들을 디스크에 백업하는 작업이다.

MMDB시스템에서는 평상시 트랜잭션 수행속도가 중요하게 여겨지기 때문에 트랜잭션 수행에 영향을 미치지 않는 체크포인팅 기법이 필요하다. 종류로는 액션 일관(action consistent)방식, 트랜잭션 일관(transaction consistent)방식, 퍼지 체크포인팅(Fuzzy checkpointing)

방식등이 있다. 액션/트랜잭션 일관 방식은 트랜잭션 수행과 동기를 맞추는 방식이며, 트랜잭션 수행에 많은 과부하를 주게 된다. 퍼지 체크포인팅 방식은 [Hagm86]에서 처음으로 제안한 방식으로 체크포인팅 프로세스가 다른 트랜잭션의 수행과 비동기식으로 갱신된 데이터가 포함된 페이지들만을 백업하는 방식으로 트랜잭션 수행과 무관하게 진행하므로 가장 우수한 시스템 성능을 가진다.
퍼지 체크포인팅은 트랜잭션과 비동기식으로 작업하므로 WAL규칙을 지킬 수 없으며, 체크포인팅을 하는 도중에 시스템이 고장나면 갱신중이던 디스크 백업 DB를 복구작업에 사용할 수 없게 된다. 시스템 성능을 위해 많은 시스템에서 퍼지 체크포인팅 방식을 채택하고 있으며, 위의 문제점들을 해결하기 위해 디스크에 두 개의 데이터베이스 백업을 가지고, 교대로 데이터베이스 백업을 체크포인팅하는 "핑퐁 갱신(ping-pong update)"을 사용한다.[JLR+94][TTT99]

● 로깅
MMDB시스템에서는 로깅이 트랜잭션 수행시 유일한 디스크 출력 작업으로 이 과정의 개선을 위한 연구로 "선완료(precommit)방식", "그룹 완료(group commit) 방식"을 들 수 있다. [Inse01]에서는 "선완료 방식"과 "그룹완료방식"을 MMDB시스템에 적용했을 때의 성능분석을 하여 "그룹완료방식"이 시스템 성능뿐만 아니라 트랜잭션 응답시간개선에도 도움이 되는 방식이며, 자료경쟁수준에 상관없이 최적의 성능을 가지는 "디스크 그룹 완료방식"을 제시하였다.

3. 디스크 기반 분산 데이터베이스시스템에서의 이중화 기법

일반적으로 데이터베이스 시스템은 성능향상을 위해 이중화기법을 사용하며, [GHO+96]에서 데이터베이스 이중화 기법을 [표 1]과 같이 두 가지 요소를 가지고 분류하였다. 첫 번째 요소는 이중화가 이루어지는 시기로 "Eager 기법"은 복사된 모든 데이터들의 갱신이 하나의 트랜잭션 완료 전에 이루어지는 것이며, 반대로 "Lazy 기법"은 트랜잭션 수행 중에는 한 지역의 데이터만 갱신되고, 이 갱신 트랜잭션이 완료된 후에 다른 지역들에 있는 복사 데이터의 갱신이 이루어지는 것이다. 전자는 일관성이 확실하게 지켜지나, 부수적인 메시지가 생기고 트랜잭션 응답시간이 길어지는 단점이 있다. 후자는 다양한 성능 향상을 가질 수 있으나, 여러 지역에 있는 복사 데이터간의 불일치가 발생한다. 이중화 분류 두 번째 요소는 갱신이 이루어지는 장소로 "Primary기법"은 "primary"라고 불리는 데이터만이 먼저 갱신되는 것으로 이중화가 단순해지나, "primary"가 있는 서버가 고장나면 전체 시스템이 중지되고, 이 지역이 성능에 병목이 되는 단점이 있다. "Update Everywhere"기법은 복사된 어느 데이터나 갱신의 대상이 될 수 있어 갱신 속도가 빨라지나, 복사 데이터간의 조정 작업이 복잡해진다[WPS+00]

update propagation		
update location	Eager Primary Cpoy	Lazy Primary Copy
	Eager Update Everywhere	Lazy Update Everywhere

[표 1] 데이터 베이스 시스템의 이중화 기법 종류

● Eager Primay Copy Replication
먼저 "primary"데이터를 갱신하고, 이어 다른 복사본들이 갱신되었다는 응답을 받으면 트랜잭션이 완료되는 방식이다. 데이터충돌은 Primary에서 조절되며, 다른 지역은 그 결정을 따르기만 하면 된다. 읽기 전용 트랜잭션은 어느 지역에서나 수행가능하며, 가장 최신의 데이터를 읽게 된다. "Primary"가 고장났을 때 바로 이어 받아 백업이 수행할 수 있도록 준비하는 "hot-standby 백업"을 구현하는 데에 쓰인다.

Phase 1:	Phase 2:	Phase 3:	Phase 4:	Phase 5:
Client Request to *Primary*	Server Coordination	Execution *Primary update*	Agreement Coordination *Apply and 2PC*	Client Response from Primary

● Eager Update Everywhere Replication *with Distributed Lock*

Phase 1:	Phase 2:	Phase 3:	Phase 4:	Phase 5:
Client Request to *any replica*	Server Coordination *Distributed Lock*	Execution *all update*	Agreement Coordination *2PC*	Client Response *from local*

● Eager Update Everywhere Replication *Based on Atomic Broadcast*

Phase 1:	Phase 2:	Phase 3:	Phase 4:	Phase 5:
Client Request to *any replica*	Server Coordination *Atomic Broadcast*	Execution *all update*	Agreement Coordinatio n	Client Response *from all*

● Lazy Primary Copy

Phase 1:	Phase 2:	Phase 3:	Phase 4:	Phase 5:
Client Request to *Primary*	Server Coordinatio n	Execution *Primary update*	Client Response from *Primary*	Agreement Coordinatio n *apply other*

● Lazy Update Everywhere

Phase 1:	Phase 2:	Phase 3:	Phase 4:	Phase 5:
Client Request to *any replica*	Server Coordination	Execution *l o c a l update*	Client Response from *local*	Agreement Coordination *apply other replicas & reconciliatio*

4. MMDB를 위한 이중화 시스템의 설계

4.1 단일 MMDB시스템의 구조

MMDB시스템은 대량으로 많은 트랜잭션들이 발생하는 분야로 길이가 짧고, 대부분 조회 위주의 트랜잭션들로 구성되며, 예측가능하고 신속한 트랜잭션 응답시간을 요구하는 시스템에 많이 적용되고 있으며, 이 요구사항에 맞추기 위하여 아래의 구조를 갖는다.
-동시성 제어 방식: 모든 트랜잭션이 DB전체에 대해 공유(shared)/독점(exclusive) 로크를 가지고 순차수행을 하며, 디스크 로깅전에 로크를 해제하는 "선완료"를 적용한다.
- 로깅의 효율화: 디스크 출력 대역폭에 맞추어 그룹완료를 수행하는 "디스크 그룹 완료"방식을 채택하여 디스크 자원 활용을 최대한으로 한다.
- 체크포인팅 방식: 트랜잭션 성능에 영향을 미치지 않으며, 신속한 복구를 위해 "핑퐁-갱신"을 하는 "퍼지 체크 포인팅"방식을 사용한다.

4.2 디스크기반 데이터베이스 시스템에서의 이중화 기법 한계점 분석

-"Eager"기법은 DB가 디스크에 있기 때문에 하나의 갱신 트랜잭션 범위 안에서 모든 복사본의 로깅이 이루어져야하기 때문에 오버헤드가 큰 "2PC(Two-Phase Commit)방식"을 이용하여 일관성을 지킨다.
- "Eager update everywhere"방식은 로킹을 사용하는 경우 많은 교착 상태를 유발시키며, 전체 시스템 성능이 크게 저하되며, Atomic Broadcasting은 "전체 순서(total-ordering)"를 구현하여야 한다.
-"Lazy"방식은 지역마다 불일치하는 데이터의 읽기를 방지하지 못한다.

4.3 MMDB이중화 시스템 설계

● 기본 동작 원리

MMDB시스템을 적용하는 응용분야에서 이중화기법을 추가하는 목적은 제공되는 응용 서비스가 시스템 에러 등과 같은 이유로 중단되지 않고 계속 진행되게 하고, 대량으로 발생하는 "읽기 전용" 트랜잭션의 정확하고, 신속한 응답 시간을 보장하기 위함이다. 또한 갱신으로 인해 전체 시스템의 성능이 저하되어서도 안된다. 정확한 복사본의 조회가 가능하게 하기 위해 "Eager"기법을 사용하면서 성능 저하를 방지할 수 있는 기법을 개발하였다. 디스크 기반 DB시스템에서는 하나의 트랜잭션 안에서 모든 복사본의 갱신과 로깅이 이루어지게 하기 위해 2PC를 사용하였으나, MMDB의 경우 DB가 메인메모리에 상주하므로 갱신이 시작된 지역만 로깅하고 다른 지역으로부터는 "Lock Ack"를 받으면 바로 트랜잭션을 완료한다. 다른 지역은 갱신을 위해 DB에 대한 "독점 로크"만 획득하면 "Lock Ack"을 보내고, 로깅은 이후에 진행시킨다. 이러한 새로운 "Eagar"기법처럼 고전적인 "Eager"방식과 같이 모든 지역에서 최신의 정확한 값을 읽을 수 있다. "Primary"기법은 모든 갱신이 하나의 서버에서만 발생하여 시스템 성능 병목이 발생하거나, 이 "primary"서버가 고장이 난 경우 시스템 전체에서 갱신이 이루어질 수 없다는 단점이 있다. 그러나, "Update everywhere"기법은 어떤 복사본도 갱신을 할 수 있게 하여 "Eager"인 경우 교착 상태가 발생하게 되고, "Lazy"인 경우 비일관성(*non-serializability*)이 발생하게 되어 이를 조절하는 "이중화 그래프(replication graph)"관리와 같은 오버헤드가 생긴다. "primary"방식의 위와 같은 단점을 없애기 위해 데이터마다 "primary"를 다르게 하여 갱신 트랜잭션을 분산시키고, 데이터마다 "primary"에 대한 "Back-up"서버를 미리 지정하여 그 데이터의 "primary"가 고장이 나더라도 즉각적으로 다른 서버가 "primary"로 동작하게 한다. 이런 "primary"기법을 적용하기 위해서는 데이터에 대한 사전 분석을 실시하여 데이터를 위치시켜 하나의 갱신 트랜잭션에는 한 서버를 "primary"로 가지는 데이터의 갱신만 있게 한다.

4.2 트랜잭션 수행과정

● 갱신 트랜잭션의 수행 과정

<"Primary"데이터 지역>

① 트랜잭션은 시작되면 고유의 "TID"를 가지고, 로컬 DB전체에 대해 "독점로크"을 획득하면 연산작업을 시작한다.

② 원하는 테이블의 레코드를 즉각적으로 갱신하고, 트랜잭션 개별 "취소 로그", "재수행 로그"를 만든다.

③연산작업이 끝나면 "TID"와 함께 모든 "재수행 로그"를 "전역 로그 버퍼"로 보내고, 동시에 네트웍을 통해 "재수행 로그"를 모든 "secondary"에게 보낸다.

④ 선완료(precommit)를 시행한다.("Mutex arry"의 로크를 획득한 후 자신이 가지고 있던 DB 독점로크를 해제한다.)

⑤"디스크 그룹 완료"가 수행되어 "재수행 로그"가 기록되었다고 디스크 컨트롤러로부터 확인을 받고, 모든 "secondary"로부터 "Lock Ack"를 받으면 트랜잭션을 완료키고, 개별 " 취소로그"를 삭제한다.

⑥ 모든 "secondary"로부터 "TID"와 함께 "Write Ack"를 받으면 그 트랜잭션의 개별 "재수행 로그"를 메인 메모리에서 삭제한다.

<"Secondary" 데이터 지역>

① "primary"시스템으로부터 "재수행 로그"를 받으면 이를 수행할 복제 갱신 트랜잭션을 만든다.

②생성된 갱신 트랜잭션은 순차적인 수행을 기다렸다가, 로컬 DB전체에 대해 "독점로크"를 가지면 "Primary"에게 "Lock Ack"를 보내고, 연산작업을 시작한다.

③ 원하는 테이블의 레코드를 갱신한 후 "재수행 로그"를 "전역 로그 버퍼"로 보내고, 선완료를 시행한다.

④ "디스크 그룹 완료"가 수행되었음을 디스크 컨트롤러로부터 확인받으면 다음 "Primary"로 보낼 메시지에 해당 "TID"와 "Write Ack"를 보낸다.

● 조회 트랜잭션

① 발생한 조회 트랜잭션은 특정 하드웨어 또는 소프트웨어에 의해 로드 밸런싱을 걸쳐 수행될 로컬 DB가 결정된다.

② DB전체 "공유 로크"를 얻으면 조회 작업을 수행한다.

③ 모든 연산작업이 끝나면 선완료를 시행한다. ("공유로크"를 해제하고, 선행되는 갱신 트랜잭션과의 일관성을 지키기 위해 "Mutex Array"의 로크를 순간적으로 획득한 후 해제한다)

④ 조회 트랜잭션을 끝낸다.

● 시스템 에러 조치 및 복구 후 과정

① 하드웨어나 "타임-아웃"등에 의해 어느 로컬 DB가 고장났음을 알게 되면 자신이 "Primary"로 동작해야 하는 데이터에 대해 이후 발생한 갱신 트랜잭션을 수행하고, 고장이 난 서버가 복구되면, 그 데이터에 대해서 다시 "Secondary"로 동작한다.

5. 결론

모든 자료가 메인 메모리에 상주하여 획기적인 트랜잭션 성능을 가져다주는 MMDB시스템에서의 더 나은 성능과 무결점을 위해 이중화 기법이 도입되고 있다. MMDB시스템에 적절한 이중화 기법을 설계하기 전에 MMDB시스템이 적용되고 있는 분야의 트랜잭션 특징과 함께 이중화 설계에 영향을 미칠 수 있는 MMDB시스템 구조에 대해 지금까지 이루어진 연구와 구현 시스템을 살펴보았다. 그리고, 지금까지 디스크 기반 분산 데이터베이스 시스템에 적용되고 있는 이중화 기법들의 종류와 그 동작 과정, 장단점들과 함께 한계사항들을 정리하였다. 이러한 고찰을 토대로 하여 현재 MMDB시스템이 적용되고 있는 분야의 트랜잭션 처리에 어떠한 제한이나 수정을 가하지 않으면서 트랜잭션 처리 능력을 배가시킬 수 있는 이중화 시스템의 기본 동작 원리와 트랜잭션 종류별 수행과정을 설계하였다. 향후에는 설계한 이중화 기법들을 실제로 MMDB시스템에 구현하여 어느 정도 성능 향상을 가져다주는 지 구체적이고 다양한 실험을 할 계획이다.

참고문헌

[Gar92] Garcia-Molina, Hector, *"Main Memory Database Systems: An Overview"*, IEEE Trans. on Knowledge and Data engineering, Vol. 4, No. 6, pp.509-516, 1992

[Hagm86], Robert Hagmann, *"A Crash Recovery Scheme for a Memory-Resident Database System"*, IEEE Transactions on Computers, C-35, 9, 1986

[Inse99] Inseon Lee,*"A Performance Evaluation of the Concurrency Control Algorithms in Main Memory Database Systems"*, In Proc. of the 26th KISS Spring Conf, April, 1999

[Inse01] Inseon Lee, *"A Study of Group Commit Policy in Main Memory Database System"*, In Proc. of the 28th KISS Spring Conf, April, 2001

[BK02] Stephen Blott, Henry F. Korth, "An Almost-Serial Protocol for Transaction Execution in Main-Memory Database Systems", In Proc. of the Int'l Conf. on Very Large Databases, 2002

[JLR+94] H. V. Jagadish, D. Lieuwen, R. Rastogi, A. Silberschatz, "Dali: A High Performance Main Memory Storage Manager", 20th International Conference on VLDB ,1994

[TTT99]The TimesTen Team, "In-Memory Data Management for Consumer Transactions The TimesTen Approach", In Proc. of the ACM SIGMOD/PIDS Int. Conf. on Management of Data, 1999

[WPS+00] M.Wiesmann, F.Pedone, A.Schiper, B.Kemme, G.Alonso, "Understanding Replication in Databases and Distributed Systems", In Proc. of the Int. Conf. on Distributed Computing Systems(ICCDS), pp 264-274, Taipei, Taiwan, April, 2000

[GHO+96] Jim Gray, Pat Helland, Patrick O'Neil, Dennis Shasha, " The Dangers of Replication and a Solution", In Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp.173-182, 1996