공유 디스크 병렬 데이타베이스에서 입출력 노드를 이용한 동시성/응집성 제어 기법

김용걸* 김양우** 진성일*** 임기욱****

A Concurrency/Coherency Control Approach using the I/O node for the Shared Disk Parallel Database

Yong Keol Kim, Yang Woo Kim, Seong Il Jin, Kee Wook Rim

〈요 약〉

병렬 데이타베이스 소프트웨어 구조 중 공유 디스크 구조는 트랜잭션간의 병렬도 향상, 적재 균형 용이, 데이타 재할당 용이 등의 장점을 가지고 있어 병렬 데이타베이스 구조 중 가장 효율적인 성능이 기대되고 있다. 그러나 공유 디스크 구조는 동시성/응집성 제어를 위한 추가적인 메세지의 증가로 네트워크 트래픽이 증가되는 문제를 가지고 있으며 이러한 문제를 완화시키고자 하는 연구가 계속되고 있다.

본 논문에서는 공유 디스크 구조의 동시성/응집성 제어를 위한 추가적인 메세지를 감소시키는 기법을 제안하고, 기존 기법과의 비교를 위해 성능 모델을 제시하였으며, 이를 통한 시뮬레이션을 수행하여 성능을 분석하였다.

* 대전보건전문대학 사무자동화과
** 동국대학교 전산통계부
*** 충남대학교 컴퓨터과학과
**** 전자통신연구소 시스템연구실

1. 서 론

멀티 프로세서 시스템의 등장과 대규모 데이타에 대한 빠른 처리 요구는 병렬 데이타베이스 시스템의 연구를 가속화시켰으며, 멀티 프로세서 시스템에 이식되는 병렬 데이타베이스 소프트웨어 구조는 병렬 데이타베이스 시스템의 성능에 매우 중요한 요인이 되고 있다.

병렬 데이타베이스 소프트웨어의 구조는 기본적으로 비공유(Shared Nothing) 구조, 공유 디스크(Shared Disk) 구조 그리고 공유 메모리(Shared Everything) 구조로 구분된다[1][2][3][4][5][6]. 지역 메모리와 지역 디스크로 구성되는 비공유 구조는 확장성이 용이한 장점을 가지나 적재 균형이 어려운 단점을 가지며, 상용화된 제품으로는 Informix의 Dynamic Scalable Architecture, Sybase의 Navigation Server, Tandem의 NonStop SQL, Teradata의 DBC/1012가 있고 연구용으로는 Arbre, Bubba, EDS, Gamma 그리고 Prisma 등이 있다. 지역 메모리와 전역 디스크로 구성되는 공유 디스크 구조는 디스크를 공유하여 적재 균형 등이 용이한 장점을 가지나 동시성/응집성 제어에 많은 비용이 소요되는 단점을 가지며 상용화된 제품으로는 IBM의 IMS, DEC의 Rdb, CODAYL의 DBMS 그리고 Oracle의 Oracle Parallel Server 등이 있다. 전역 메모리와 전역 디스크로 구성되는 공유 메모리 구조는 적재 균형이 가장 용이한 장점을 가지나 가용도가 낮고 확장이 어려운 단점을 가진다.

공유 디스크 구조는 트랜잭션간의 병렬도 향상, 적재 균형 용이, 데이타 재할당 용이 등의 장점을 이용하여 가장 효율적인 병렬 데이타베이스 시스템의 구조로 활용될 수 있으나[6], 동시성/응집성 제어를 위한 추가적인 메세지들로 통신 오버헤드가 증가되어 시스템의 성능이 저하되는 문제점을 가지고 있다. 기존의 동시성/응집성 제어 기법[7][8][9][10][11][12][13]에서는 이러한 오버 헤드를 많이 완화한 기법으로 하나의 메세지를 이용하여 처리될 페이지의 동시성과 응집성을 검사하는 기법이 사용되고 있으나 디스크 요구를 위한 추가적인 메세지가 발생하는 문제점을 해소하지는 못하고 있다.

본 논문에서는 공유 디스크 구조에서의 단점인 동시성/응집성 제어 및 디스크 접근 요구를 위해 전송되는 메세지 수를 감소시켜 공유 디스크 구조에서의 통신 부담을 완화하기 위해 입출력 노드(Input Output Node)를 이용한 동시성/응집성 제어 기법을 제시한다. 또한 기존 기법과의 비교를 위해 각 구조에 대한 성능 모델을 작성하여 시뮬레이션을 수행한 후, 그 결과를 분석함으로써 두 기법간의 비교 · 분석을 수행하였다.

본 논문의 구성은 다음과 같다. 2장에서는 동시성/응집성 제어 기법의 배경에 대해 설명하고, 3장에서는 본 논문에서 제시하는 입출력 노드를 활용한 동시성/응집성 제어 기법을 설명하며, 4장에서는 각 기법들의 성능 모델을 제시하고, 5장에서는 성능 분석에 대해 기술한다. 마지막으로 6장에서는 결론을 다룬다.

2. 관련 연구

공유 디스크 구조의 병렬 데이타베이스 시스템에서의 동시성/응집성 제어 기법은 다음과 같다.

2.1 중앙 로킹 기법(Central Locking Approach)

중앙 로킹 기법에서는 특정한 노드에서 수행

되는 하나의 전역 로크 관리자(Global Lock Manager:GLM)가 전체적인 동시성 제어를 수행하게 된다. 전역 로크 관리자는 모든 노드에서 발생하는 로크 요구와 해제를 처리하고, 이를 위한 전역 로크 테이블을 관리한다. 모든 노드들은 로크 요구나 해제 요구가 있을 경우 전역 로크 관리자가 있는 노드에게 메세지를 통해 요구한다. 중앙 로킹 기법은 프로토콜의 단순성 때문에 구현과 교착상태 감지가 용이한 장점을 가지나 전역 로크 관리자에게 모든 로크 요구 및 해제 요구가 집중되어 병목현상이 발생하는 단점을 가진다.

응집성 제어를 위한 여러 기법인 Check-on-Access기법은 전역 로크 관리자를 사용하는 로킹 프로토콜에 적용할 수 있다. 로크 테이블의 정보를 확대해서, 전역 로크 관리자가 로크 요구를 처리할 때 버퍼 페이지가 유효한 지를 점검할 수 있도록 하는 것이다. 로크는 페이지가 접근되기 전에 얻어야 하므로 로크 요구시 무효화된 페이지 복사본들이 검사가 가능하므로 추가적인 메세지 전송이 필요 없다. 무효화된 버퍼를 점검하기 위해서 필요로 하는 정보는 쓰기 로크의 해제와 함께 통신의 추가적인 메세지 없이 갱신될 수 있다. 단점은 무효화된 페이지가 나중에 제거됨으로 해서 히트율이 떨어질 수 있다는 것이다[10].

2.2 분산 로킹 기법 (Distributed Locking Approach)

전체 데이타베이스의 로크 정보를 한 노드에서 관리하여 병목현상이 발생하는 중앙 로킹 기법의 문제를 해결하기 위해, 분산 로킹 기법은 전체 데이타베이스에 대해 로크 영역을 나누어 각 노드가 해당 데이타베이스 영역에 대해서만 로킹을 관리하도록 로크 관리자를 전 노드에 분산시켰다. 각 노드는 각 로크를 어느 노드에서 관리하는 지에 대한 정보를 가지고 있는 것으로 가정한다. 따라서 로크 요구 메세지나 해제 메세지는 해당 로크 관리자를 가진 노드로 보내진다. 이 방식은 여러 노드에 로크와 통신의 부하가 분산됨으로써 시스템의 성능이 높아질 수 있는 장점을 가진다. 반면에 로크를 여러 노드로부터 얻었을 경우 다수의 해제 메세지가 필요하게 되고, 로크에 대한 정보도 분산되어 있으므로 교착상태를 파악하기 힘들고 해결하는 방법이 복잡해지는 단점을 가진다.

응집성 제어는 중앙 로킹 기법에서 설명한 Check-on-Access기법을 사용하고 있으며 처리될 페이지를 다른 노드에 요구하기 위하여 로크 요구 메세지를 전송한다. 만일 로크를 요구하는 페이지가 버퍼에 존재하면 이 페이지의 일련 번호를 로크 요구 메세지에 포함시킨다. 각 노드는 다른 노드로부터의 로크 요구 메세지를 받으면 해당 페이지의 로크를 검사하여 로크를 획득하면 해당 페이지가 버퍼에 있는지를 검사한다. 만일 버퍼에 존재하면 버퍼에 있는 페이지의 일련번호를 로크 요구 메세지에 포함된 일련번호와 비교하여 최근의 페이지가 아니면 로크 응답 메세지와 함께 해당 페이지를 같이 전송한다. 만일 요구한 노드가 최근의 페이지를 유지하고 있는 경우에는 최근의 페이지를 보내지 않고 로크 응답 메세지만을 전송한다.

2.3 디스크 제어기 로킹 기법 (Disk Controller Locking Approach)

로크 요구 메세지를 없애기 위하여 디스크 제어기에 전역 로크 관리자를 유지하는 기법이다[3]. 로크 관리자는 여러 디스크 제어기에 분산되어 할당된 데이타베이스만을 관리한다. 분산 로킹

기법과 유사하나 분산 로킹 기법은 로크 관리자가 프로세싱 노드(processing node)에 존재하고, 이 기법은 로크 관리자가 디스크 제어기에 있다는 차이를 가지고 있다. 트랜잭션을 처리하는 노드에서 처리할 페이지의 로크를 검사하기 위하여 로크 요구 메세지를 해당 페이지의 로크를 관리하는 디스크 제어기에 보낸다. 디스크 제어기에서는 해당 로크를 검사하고 로크를 획득하면 디스크를 구동 시켜 해당 페이지를 읽어 요구한 노드에 전송한다.

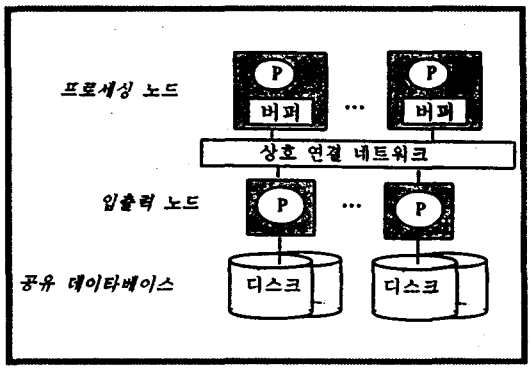
디스크 제어기 로킹 기법의 로크 관리자는 동시성 제어만을 고려하였으며 응집성 제어를 위해 로크 관리자와 관계없이 프로세싱 노드에서 버퍼 무효화 프로토콜을 사용해야 하는 부담을 가지고 있다[3].

이 기법은 로크 요구를 위한 메세지가 없는 장점을 가지나 응집성 제어를 위한 페이지를 접근할 때마다 추가적인 메세지가 발생되는 부담을 가지고 있고, 버퍼 히트인 경우에도 디스크 제어기에 메세지를 전송해야 하고, 로크 관리자를 위한 특수한 하드웨어를 구현해야 하는 단점을 가지고 있다.

3. 새로운 동시성/응집성 제어 기법

3.1 시스템 구조

본 논문에서 가정한 시스템 구조는 [그림 1]과 같다. 프로세싱 노드는 하나의 프로세서와 지역 메모리를 가지고 있으며, 입출력 노드도 입출력을 담당하는 전용 프로세서와 지역 메모리로 구성된다. 각 프로세싱 노드와 각 입출력 노드와의 통신을 위해 통신 네트워크가 존재한다. 이러한 구조의 병렬 처리 시스템은 상호 연결 네트워크를 중심으로 동일한 인터페이스의 처리 노드와 입출력 노드를 갖는 구조로 계산 기능과 입출력 성능 간의 성능 분배가 유연한 구조이며 현재 전자통신연구소에서 개발 중인 SPAX[13]와 IBM의 SP2[14] 등이 이러한 구조를 갖는다. 입출력 노드는 프로세싱 노드의 프로세서가 입출력에 직접적으로 관여하는 것을 방지하여 입출력 노드가 입출력을 수행하는 동안 다른 트랜잭션을 처리할 수 있어 프로세서의 효율성을 높이기 위해 사용하는 일반적인 구조이다.



〈그림 1〉 시스템 구조

입출력 노드는 디스크 입출력을 전담하여 처리하는 노드로 프로세싱 노드의 입출력 요구에 따라 디스크 드라이버를 구동하여 디스크에서 읽어들인 데이타를 요구한 프로세싱 노드로 전달하는 기능을 담당하는 입출력 전용 노드이다. 입출력 노드는 지역 메모리, 프로세서로 구성되며 프로세싱 노드와 입출력 노드와의 통신은 메세지 전송 방식에 의해 이루어 진다. 또한 입출력 노드는 [그림 2]와 같이 서버 프로세스가 입출력 노드 내에 존재하여 프로세싱 노드에서의 모든 입출력 요구를 처리하도록 구성된다. 입출력 노드 커널은 메모리에 상주하여 프로세스 관리 및 입출력 관리 기능을 제공한다.

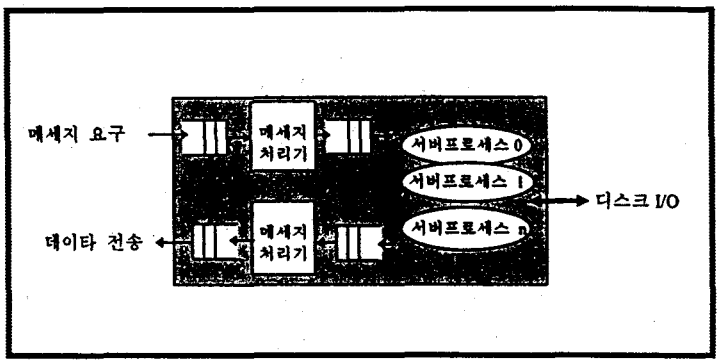
3.2 제안 기법

본 논문에서 제안한 입출력 노드 로킹 기법(Input Output Node Locking Approach)을 위한 입출력 노드는 [그림 3]과 같다.

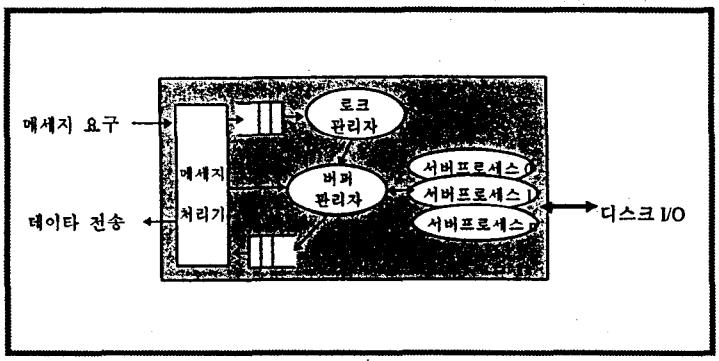
병렬 데이타베이스 시스템은 트랜잭션 관리자, 버퍼 관리자, 로크 관리자, 통신 관리자, 스케줄러 그리고 메타 데이타 관리자의 상호 작용을 통하여 트랜잭션을 처리한다. 각 모듈은 서로 독립적으로 수행 가능한 모듈로 구성되어 있다.

특히 동시성 제어를 담당하는 로크 관리자는 트랜잭션이 처리할 데이타의 일치성을 보장해 주기 위한 기능을 제공하기 위하여 로크 테이블을 유지하여 어떤 한 트랜잭션이 접근하고자 하는 페이지가 다른 트랜잭션에 의해 사용되고 있으면 대기 큐에서 사용이 완료되기를 기다리고 있다가 로크가 해제되면 해당 데이타의 로크 권한을 부여하는 기능을 담당한다. 이러한 로크 관리자의 기능 요구는 IPC(Inter Process Communication) 기법이나 직접적인 함수 호출에 의해 이루어지고 있다.

또한 로크 관리자에서 사용하는 로크 테이블은 다른 모듈과는 독립적으로 운영·유지되고 있다. 입출력 노드는 독자적인 운영체제에 의해 동작되므로 이러한 로크 관리자의 이식이 가능하다.

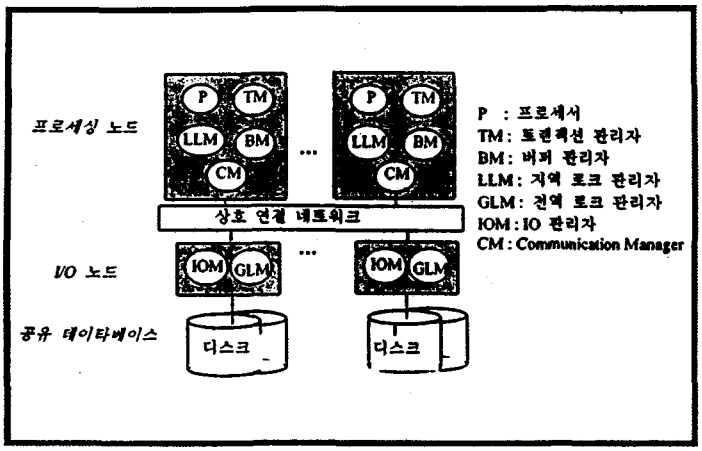


〈그림 2〉 입출력 노드의 명령 흐름도



〈그림 3〉 입출력 노드에서의 로크 관리자

이러한 로크 관리자의 운영을 위해서는 프로세싱 노드에서 로크 요구 메세지에 디스크 접근 정보가 포함될 수 있어야 한다. 또한 메세지 처리기는 입출력 요구 메세지인지 로크 요구 메세지인지를 구분하는 기능을 가진다. 만일 입출력 요구 메세지인 경우에는 버퍼 관리자에게 해당 페이지가 버퍼에 있는지 검사하고 버퍼에 있으면 바로 페이지를 전송하고 그렇지 않으면 디스크를 구동하는 프로세스에 해당 메세지를 전달하여 페이지를 읽어 들여 페이지를 요구한 노드에 전송한다. 로크 요구 메세지인 경우에는 로크 관리자에게 로크를 획득한 후 버퍼 관리자에게 해당 페이지를 요구한다.



〈그림 4〉 입출력 노드 로킹 기법

본 논문에서는 Zipf의 법칙[3]을 적용하여도 대용량의 데이타베이스를 고려할 때 버퍼 히트율이 낮아지고 디스크 입출력이 자주 발생되는 점을 고려하여 입출력 노드 로킹 기법을 제안 하였으며 [그림 4]와 같은 구조를 가진다. 각 노드는 원하는 페이지의 로크 권한을 가진 입출력 노드에 대한 정보를 유지하는 것으로 가정한다.

입출력 노드 로킹 기법을 위한 수행 절차는 [그림 5]와 같다. 각 노드에 있는 지역 로크 관리자는 읽기 최적화(Read Optimization)기법에 따르는 구조로 각 노드가 페이지에 대한 읽기 로크를 요구하는 경우 전역 로크 관리자는 해당 페이지의 로크 권한을 지역 로크 관리자에게 넘겨주고 이러한 페이지들을 버퍼에 위치시켜 관리한다. 입출력 노드에 위치한 GLM은 자신이 관리하는 디스크 내의 페이지들에 대한 로크 권한을 가진다.

TM은 트랜잭션의 읽기 요구가 발생한 경우 해당 노드의 LLM에 로크 권한이 있는 지를 검사하여 로크 권한을 가진 경우에는 로크를 획득하고 BM을 통해 버퍼에 있는 페이지를 접근하여 처리한다. 만일 로크 권한이 LLM에 없는 경우에는 로크 권한을 가진 입출력 노드에 로크 요구 메세지를 CM을 통해 전송한다. 입출력 노드는 읽기 로크 요구 메세지에 따른 페이지와 로크 권한을 해당 노드에 전송한다. 트랜잭션의 쓰기 요구가 발생하면 TM은 쓰기 로크 요구 메세지를 CM을 통해 입출력 노드에 전송한다. 입출력 노드의 GLM은 해당 페이지의 로크를 검사하여 쓰기 로킹이 있는 경우에는 해제될 때까지 기다리게 되지만 이미 읽기 로킹이 있는 경우에는 읽기 로크 권한을 취소한 후에 해당 페이지를 접근한다.

처리 노드에서 :

1. 페이지의 읽기 요구인 경우,
   1.1 지역 로크 관리자로 로크 요구 메세지 발생.
   1.2 지역 로크 관리자에 페이지 권한이 있는 경우, 로크 응답 메세지 발생.
   1.3 없는 경우, 입출력 노드의 전역 로크 관리자로 로크 요구 메세지 전송
   1.4 응답 메세지 기다림
2. 페이지의 쓰기 요구인 경우
   2.1 입출력 노드의 전역 로크 관리자로 로크 요구 메세지 전송
   2.2 응답 메세지 기다림
3. 트랜잭션 완료의 경우
   3.1 읽기 요구인 페이지에 대해, 로크 해제 요구 메세지 전송
   3.2 쓰기 요구 페이지에 대해, 로크 해제 요구 메세지와 페이지 함께 전송

입출력 노드에서:

1. 로크 요구 메세지인 경우
   1.1 페이지 버전 번호가 포함되어 있는 경우
      1.1.1 로크 관리자 내의 페이지 버전 값과 같으면, 로크 응답 메세지만 전송
      1.1.2 다르면,
         1.1.2.1 요구된 페이지가 버퍼에 있으면, 로크 응답 메세지와 요구된 페이지 함께 전송
         1.1.2.2 없으면, 디스크에서 해당 페이지를 읽은 후, 로크 응답 메세지와 요구된 페이지 함께 전송
2. 로크 해제 요구 메세지
   2.1 읽기 요구인 경우, 로크 해제
   2.2 쓰기 요구인 경우, 해당 페이지의 버전 번호 변경, 로크 해제

〈그림 5〉 입출력 노드 로킹 기법의 수행 절차

3.3 기존 기법과의 비교

입출력 노드 로킹 기법은 기존의 분산 로킹 기법과 디스크 제어기 로킹 기법을 혼용하여 각 기법들의 단점을 보완하였으며 각 기법과의 비교는 다음과 같다.

노드 B가 페이지 P의 접근을 요구하고 P는 디스크 X에 있고, IOX는 디스크 X의 입출력 노드이다. 현재 로크 권한은 노드 C에 있다. 또한 임의의 한 노드를 제외한 다른 노드 AN이 있다고 가정한다. 이때 메세지 수는 M으로 입출력을 위한 메세지 수는 IO로 가정할 때 통신 메세지의 수와 입출력 요구에 대한 메세지 전송 수에 따른 비교표는 [표 1]과 같다. 디스크 제어기 로킹 기법에서는 응집성 제어를 위해 버퍼 무효화 프로토콜을 이용한다. 즉, 한 트랜잭션 완료 시에 현재 노드에서 변경된 모든 페이지에 대해 다른 노드로 무효화 메세지를 전송해야 하는 부담을 가진다.

〈표 1〉 통신 메세지와 입출력 수에 따른 비교표

	입출력 노드 로킹	분산 로킹	디스크 제어기 로킹
P가 B의 버퍼에 있는 경우(Read 로크)	B{ReqLock,Pid} M:0, IO:0	B{ReqLock, Pid} M:0, IO:0	B->X{ReqLock,Pid} X->B{AckLock,Pbuf} M:2, IO:0
P가 B의 버퍼에 있는 경우(write 로크)	B->IOX{ReqLock,Pid} IOX->B{AckLock,Pid} 트랜잭션 완료후; B->IOX{Bufinv,Pbuf} M≒2, IO:0	B{ReqLock, Pid} M:0, IO:0	B->X{ReqLock,Pid} X->B{AckLock,Pbuf} 다른 각 노드에 대해; B->AN{Bufinv,Pbuf} M〉3, IO:0
P가 C의 버퍼에 있는 경우, 또는 P가 IOX의 버퍼에 있는 경우	B->IOX{ReqLock,Pid} IOX->B{AckLock,Pid} 트랜잭션 완료후; B->IOX{Bufinv,Pbuf} M≒2, IO:0	B->C{ReqLock,Pid} C->B{AckLock,Pid} 트랜잭션 완료후; B->C{Bufinv,Pbuf} M≒2, IO:0	B->X{ReqLock,Pid} X->B{AckLock,Pbuf} 다른 각 노드에 대해; B->AN{Bufinv,Pbuf} M〉3, IO:0
P가 버퍼에 없는 경우	B->IOX{ReqLock,Pid} IOX{Read,Pid} IOX->B{AckLock,Pid} 트랜잭션 완료후; B->IOX{Bufinv,Pbuf} M≒2, IO:1	B->C{ReqLock,Pid} C->X{Read,Pid} X->C{Pbuf} C->B{AckLock,Pid} 트랜잭션 완료후; B->C{Bufinv,Pbuf} M≒4, IO:1	B->X{ReqLock,Pid} X{Read,Pid} X->B{AckLock,Pbuf} 다른 각 노드에 대해; B->AN{Bufinv,Pbuf} M〉3, IO:1

[표 1]에서와 같이 전체적으로는 분산 로킹 기법이 전송되는 메세지의 수가 작은 것으로 비교되나, 대용량 데이타베이스를 가정으로 하는 병렬 데이타베이스에서의 버퍼 히트율이 30-40%[15][16]정도로 낮은 점을 고려하면 분산 로킹 기법에 비해 입출력 노드 로킹

기법에 의해 발생되는 메세지 수가 감소되어 시스템의 성능이 향상될 것으로 예상된다.

입출력 노드 로킹 기법의 장단점은 다음과 같이 요약할 수 있다. 이 기법의 장점은 첫째, 하나의 메세지를 이용하여 로크 검사, 버퍼 무효 검사뿐만 아니라 입출력 요구까지 처리되어 전송되는 메세지의 수가 감소된다는 것이고, 둘째는 멀티미디어 데이타, 의사 결정 지원(Decision Support System)형 질의어와 같이 읽기 연산 중심이고, 버퍼 히트율이 매우 낮은 경우에 매우 효율적이다. 세째는 데이타 재할당을 위한 추가적인 부담을 줄일 수 있다. 또한 입출력 노드 로킹 기법의 단점은 쓰기 연산인 경우에 버퍼 히트에 관계없이 로크 요구를 위해 입출력 노드로 메세지를 전송한다는 것이다.

4. 성능 모델

본 장에서는 공유 디스크 구조에서의 동시성/응집성 제어를 위한 분산 로킹 기법과 본 논문에서 제안한 입출력 노드 로킹 기법간의 성능을 비교하기 위해 각 기법의 성능 모델을 제시한다. 성능 모델을 위한 시스템의 성능 매개 변수는 참고문헌[7][11]에서 참조하였다.

4.1 성능 매개 변수

공유 디스크 구조의 성능 분석을 위한 성능 매개 변수는 아래의 [표 1]과 같이 정의한다. 데이타베이스 페이지 크기는 4KB이고, 동시성 제어를 위해 전송되는 메세지와 디스크 접근 요구 메세지의 크기는 100B로 하였으며 메세지를 처리하기 위한 명령어 수는 5,000개로 정하였다. 각 노드에 있는 프로세서의 처리속도는 100MIPS로 정하고 각 디스크의 대역폭은 초당 10 MB로 하였으며, 네트워크의 초당 전송속도는 100MB로 정하였다. 프로세싱 노드와 입출력 노드 갯수는 기본값을 4개로 정하였고 최대 20개까지 확장할 수 있다. 한 트랜잭션이 처리하는 페이지 갯수는 평균 10개를 접근하여 처리하는 것으로 하였고, 질의 발생 수는 초당 2,000개의 트랜잭션을 발생하

〈표 2〉 성능 매개변수

매개변수명	값	매개변수명	값
페이지의 크기	4Kbytes	페이지당 명령어수	평균12,500개
메세지의 크기	100 bytes	다중 프로그래밍도	50
메세지당 명령어수	5000명령/메세지	버퍼 히트율	40%
CPU속도	100 MIPS	디스크 캐쉬 히트율	30%
디스크 대역폭	10 Mbytes	로크 충돌 확률	1%
네트워크 대역폭	80~160 Mbytes/초	페이지 읽기 요구	70%
프로세싱 노드수	4~16개	페이지 쓰기 요구	30%
입출력 노드 수	4~16개	로컬 데이타 접근율	70%
처리될 페이지수	평균 10개/트랜잭션	원격 데이타 접근율	30%
질의 발생 수	평균1,000~ 5,000개/초(지수분포)	평균 로크 지연시간	평균0.15msec(지수분포)

도록 하였으며, 하나의 페이지를 처리하기 위한 명령어 수는 평균 12,500개로 하였다.

시스템 내에서 활성(active) 상태인 트랜잭션의 수를 노드당 50개로 제한하였다. 프로세싱 노드에서의 버퍼 히트율은 40%로 그리고 입출력 노드에서의 캐쉬 히트율은 30%로 정하였고, 페이지 요구를 한 경우에 해당 페이지가 로크되어 있을 확률은 읽기 요구나, 쓰기 요구에 대해 모두 1%로 정하였다. 트랜잭션의 페이지 요구중 70%는 읽기 요구를 30%는 쓰기 요구를 하는 것으로 하였다. 또한 트랜잭션이 요구하는 페이지의 로크 권한이 자신의 노드일 확률을 70%로 정하였고 다른 노드에 있을 확률을 30%로 정하였다.

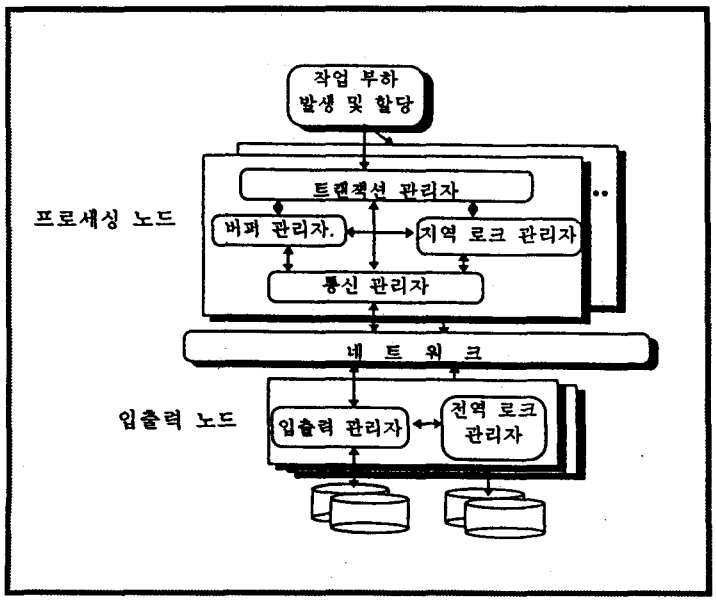
4.2 성능 모델

4.2.1 공유 디스크 구조를 위한 성능 모델

트랜잭션 관리자는 처리될 트랜잭션을 받아 처리될 페이지가 있으면 해당 페이지의 처리를 위해 로크를 검사한다.

이때 처리할 페이지가 다른 노드에 존재하는 경우에는 해당 노드로의 페이지 로크를 획득하기 위해 통신 관리자를 통해 다른 노드에 로크 요구 메세지를 보내고 이에 대한 응답을 받을 때까지 기다린다. 자신의 노드에 로크 권한이 존재하는 경우에는 자신의 노드에 있는 로크 관리자에게 로크 검사를 요구하고 기다린다. 로크를 획득한 경우에 해당 페이지가 버퍼에 있으면 즉시 접근하여 처리한 후 다음 페이지 접근을 트랜잭션 관리자에 요구한다.

만일 디스크에 존재하는 경우에는 디스크에 페이지 입출력 요구 메세지를 통신 관리자를 통해 요구한 후 페이지가 버퍼로 전송될 때까지 기다린다. 트랜잭션 관리자는 트랜잭션들이 페이지 처리를 완료하면 각 트랜잭션이 소유한 페이지들의 커미트 요구 메세지를 해당 로크 관리자에게 전송한 후 커미트한다. 입출력 관리자는 디스크 접근 요구를 감지하고 입출력 노드 버퍼에 있으면 해당 노드로 페이지를 전송하고, 만일 버퍼에 존재하지 않으면 디스크를 구동시켜 페이지를 읽어 들인 후 해당 노드로 전송한다[그림 6].



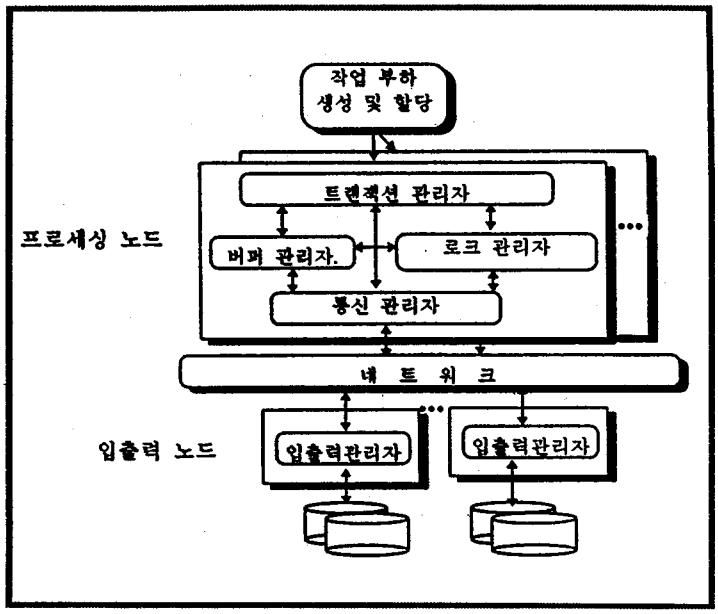
〈그림 6〉 분산 로킹 기법을 위한 성능 모델

4.2.2 입출력 노드 로킹 기법을 위한 성능 모델

트랜잭션 관리자는 처리될 트랜잭션을 받아 처리될 페이지가 있으면 해당 페이지의 처리를 위해 로크를 검사한다. 읽기

요구인 경우에는 지역 로크 관리자에 로크를 요구한다. 만일 로크를 획득한 경우에는 버퍼에 페이지가 존재하는 경우이므로 즉시 페이지를 처리하나, 페이지가 없는 경우에는 입출력 노드에 디스크 접근을 요구한다.

이때 로크 요구 메세지는 페이지 접근 정보를 포함하여 입출력 노드의 로크 관리자에게 통신 관리자를 통해 보내고 이에 대한 응답을 받을 때까지 기다린다. 입출력 관리자는 로크 요구 메세지를 받은 경우에는 로크 관리자에게서 로크를 획득한 후 해당 페이지가 버퍼에 있으면 해당 노드로 전송하고 디스크에 있는 경우에는 바로 디스크를 구동하여 페이지를 읽어들인 후 해당 노드로 전송한다[그림 7].



〈그림 7〉 입출력 노드 로킹 기법을 위한 성능 모델

5. 성능 분석

성능 모델은 객체 지향 시뮬레이션 언어인 DEVSIM++[17]를 이용하여 구현하였다. 시뮬레이션 수행 시간은 10초로 하였으며 성능 분석을 위해 시스템의 처리량, 응답 시간, 네트워크 지연 시간, 그리고 디스크 대기시간을 이용하여 두 가지 동시성/응집성 제어 기법에 대한 비교 · 분석 하였다.

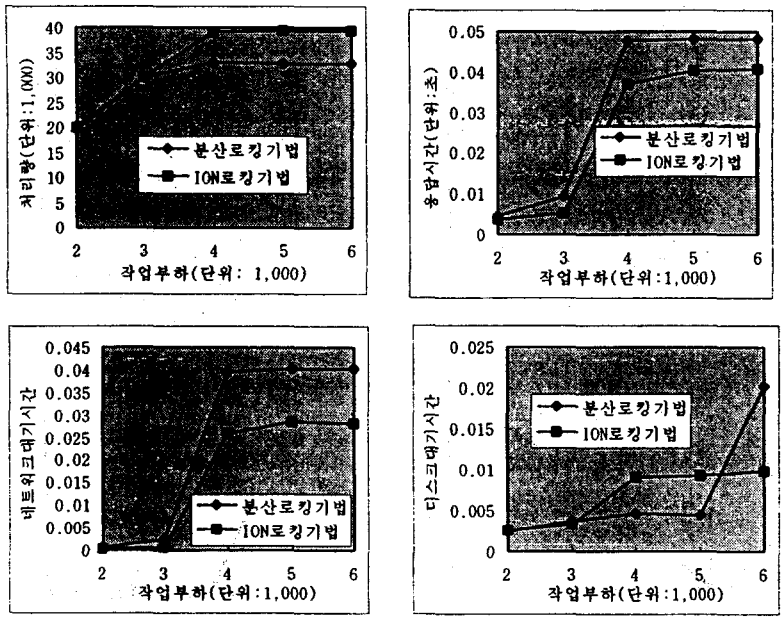
5.1 작업 부하에 따른 변화

[그림 8]은 작업 부하에 따른 변화를 도표화하였으며 프로세싱 노드 수를 4개, 입출력 노드 수는 7개로 구성하였을 때의 작업 부하에 따른 실험 결과이다. 초당 3,000개의 트랜잭션이 발생한 경우의 처리량은 두개의 기법이 같으나 초당 4,000개인 경우에는 분산 로킹 기법은 32,000개의 처리량을 보인 반면에 입출력 노드 로킹 기법은 40,000개의 처리량을 보이고 있다. 제안 기법이 약 25%의 성능 향상을 보이고 있다.

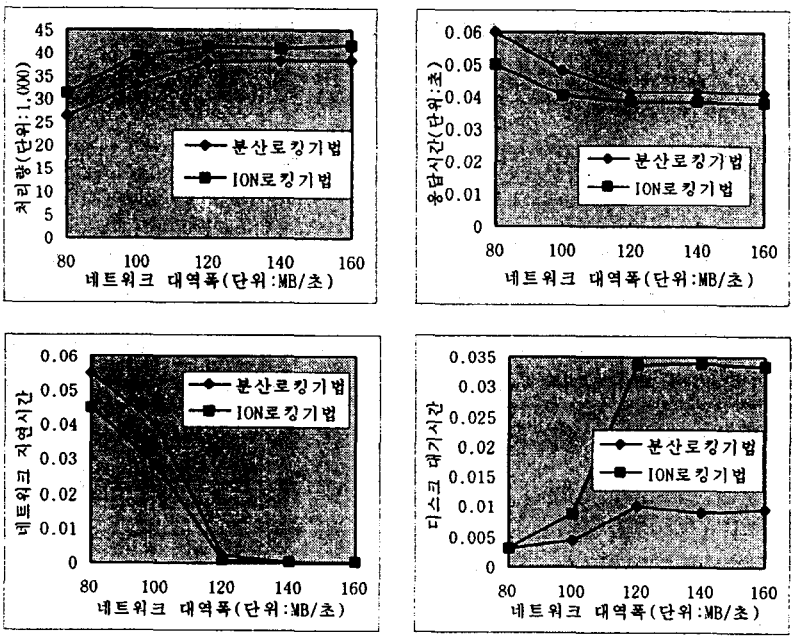
제안 기법은 작업 부하의 증가에 따라 처리율이 일정한 이유는 네트워크의 트래픽이 크게 증가하기 때문으로 분석된다.

5.2 네트워크 대역폭에 따른 변화

[그림 9]는 네트워크 대역폭의 변화에 따른 시스템의 영향을 분석한 도표이다. 초당 120MB인 경우가 가장 좋은 성능을 보이고 있으나 그 이상인 경우에는 디스크 대기시간의 증가로 비슷한



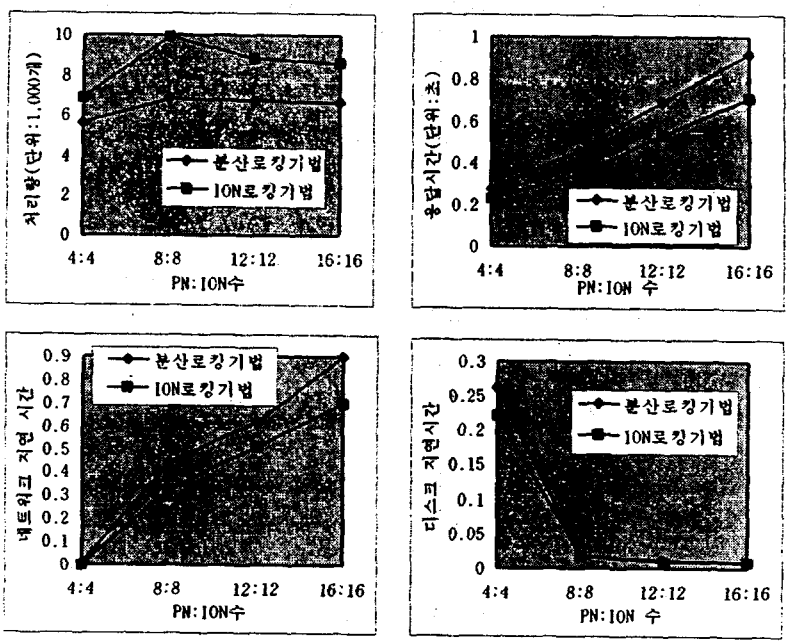
〈그림 8〉 작업 부하의 변화



〈그림 9〉 네트워크 대역폭의 변화

처리량을 보이고 있다.

전반적으로 입출력 노드 로킹 기법은 분산 로킹 기법에 비해 약 10%이상의 성능 향상을 보이고 있다. 제안 기법은 네트워크의 대역폭의 증가에 따라 입출력 대기 시간이 크게 증가하여 처리율에 많은 영향을 주고 있다.



〈그림 10〉 의사 결정 지원형 트랜잭션의 변화

5.3 의사 결정 지원형 트랜잭션의 변화

공유 디스크 구조의 장점인 의사 결정 지원형 트랜잭션의 처리에 관한 실험으로 한 트랜잭션당 처리하는 페이지 수를 50개로 설정하였고, 트랜잭션의 페이지 요구는 90%가 읽기 요구와 10%의 쓰기 요구인 경우로 가정하였고, 작업 부하는 초당 1,000개의 트랜잭션이 발생하는 것으로 가정하였다.

[그림 10]은 이에 대한 결과를 도표화하였으며 8:8인 경우에 가장 좋은 성능을 보이고 있으며, 분산 로킹 기법의 경우보다 입출력 노드 로킹 기법이 약 60%정도의 성능 향상이 있는 것으로 나타났다.

8:8이상인 경우에는 노드의 수가 많아질수록 네트워크 대기시간의 증가로 전체 시스템의 성능이 오히려 감소되고 있다.

6. 결론

공유 디스크 구조에서의 동시성/응집성 제어 기법은 추가적인 메세지를 유지해야 하고 이러한 추가적인 메세지로 인하여 네트워크 트래픽을 증가시켜 시스템의 성능을 저하시키는 요인이 되고 있다. 이러한 문제점을 해결하기 위한 기존의 기법은 동시성/응집성을 하나의 메세지로 처리할 수 있는 방법을 제공하고 있으나 추가적인 입출력 요구 메세지를 요구하고 있고 기존의 동시성/응집성 제어를 위한 시스템 구조로는 로크 요구, 버퍼 무효화 검사 요구, 그리고 입출력 요구를 하나의 메세지를 통해 해결할 수 없다.

본 논문에서는 이러한 동시성/응집성 제어를 위한 추가적인 메세지를 감소시켜 시스템의 성능을 향상하기 위해 로크 관리자를 프로세싱 노드에

서 관리하지 않고 입출력 노드에서 관리하는 입출력 노드 로킹 기법을 제안하였다. 또한 성능 모델을 통한 시뮬레이션을 수행하여 성능을 분석하였다. 분석된 결과로는 제안한 입출력 노드 로킹 기법이 동시성/응집성 제어를 위한 메세지 수의 감소로 인하여 노드 수의 변화, 트랜잭션 형태의 변화 그리고 네트워크 대역폭을 변화시킴에 따라 트랜잭션의 처리율이나 응답 시간에 있어서 기존 분산 로킹 기법보다 약 10~60%정도의 성능 향상이 있음을 보였다.

〈참고문헌〉

[1] David DeWitt, Jim Gray, Parallel Database Systems: The Future of High Performance Database Systems, *Communication of ACM*, Vol. 35, No. 6, pp. 85-98, June 1992.

[2] H. Pirahesh, C. Mohan, J. Cjeng, T.S. Liu,P. Selinger, Parallelism in Relational DataBase Systems: Architectural Issues and Design Approaches, *Second Intl Symp. Databases in Parallel and Distributed Systems*, pp.4-29, 1990.

[3] Anupam Bhide, An Analysis of Three Transaction Processing Architecture, *Proc. 14th Intl Conf. Very Large DataBases*, pp.339-350, 1988.

[4] 김양우, 박진원, 임기욱, 병렬 DBMS 아키텍춰 분석, *정보과학회지*, 제13권,제7호 pp. 60-69,1995.7,.

[5] Mohan C., etc., Parallelism in relational database management systems, *IBM Systems Journal*, Vol. 33, No. 2, pp.349-371, 1994.

[6] Ehard Rahm, Parallel Query Processing in Shared Disk Database Systems, TR 1/93, Univ. Leipzig, Dept. of Comp. Science.

[7] S.H. Kim,Y.K. Kim,Y.K. Park,S.I. Jin,Y.W. Kim,J.W. Park,K.O. Lim,"The Performance Evaluation of Parallel Cluster System Structures for Database Application," *Journal of Mathematical Modeling and Science Computing*, Vol. 6, Apr. 1996.

[8] Mohan C., Narang I., Recovery and Coherency-Control Protocols for Fast Intersystem Page Transfer and Fine-Granularity Locking in a Shared Disks Transaction Environment, *Proc. 17th International Conference on Very Large Data Bases*, Barcelona,pp.193-207, Sept. 1991.

[9] Mohan C., Narang I., Efficient Locking and Caching of Data in the Multisystem Shared Disk Transaction Environment, *Proc. 3rd International Conference on Extending Database Technology, Vienna, March 1992,also IBM Research Report RJ 8301*, Aug. 1991.

[10] Ehard Rahm, Concurrency and Coherency Control In Database Sharing Systems, TR ZRI 3/91, Dec, 1991

[11] Ehard Rahm, Empirical Performance of Concurrency and Coherency Control Protocols for Database Sharing

Systems, *ACM Transaction on Database Systems*, Vol. 18, No. 2, pp. 333-377, June 1993.

[12] Dias D.M., Iyer B.R., Robinson J.T., Yu P.S., Integrated Concurrency-Coherency Controls for Multisystem Data Sharing, *IEEE Trans. Softw. Eng.*, 15, 4, pp.437-448 ,1989.

[13] Y.W. Kim, S.W. Oh, J.W. Park, Design Issues and System Architecture of TICOM-IV, A Highly Parallel Commericial Computer, *The 3rd Euromicro Workshop on Parallel & Distributed Processing*, San Remo, Italy, pp.219-226, Jan. 1995.

[14] T. Agarwala, J.L. Martin, et al,SP2 System Architecture, *IBM System Journal*, Vol. 34, No. 2 pp. 152-183, 1995,.

[15] Dan A., Dias D.M., Yu P.S., The Effect of Skewed Data Access on Buffer Hits and Data Contention in a Data Sharing Environment, *In Proceeding of the 16th International Conference on Very Large DataBases*, Brisbane pp.419-431,Aug. 1990.

[16] A. Dan, P.S. Yu, Performance Comparisons of Buffer Coherency Policies, *IEEE*,pp.208-217,1991.

[17] Tag Gon Kim, DEVSIM++ Users Manual Ver. 1.0, Computer Engineering Research Lab., Dept. of Electrical Eng., KAIST,1994.