# 공유 디스크 클러스터 기반의 실시간 트랜잭션 처리 알고리즘 성능 평가

이상호° 온경오 조행래
영남대학교 컴퓨터공학과
{comman35°, ondal}@yumail.ac.kr, hrcho@yu.ac.kr

# Performance Evaluation of Real-Time Transaction Processing Algorithms in Shared Disks Clusters

Sangho Lee° Kyungoh Ohn Haengrae Cho
Computer Science Engineering, Yeungnam University

**요 약**

인터넷을 이용한 전자 상거래 및 관리 시스템 등의 실시간 처리를 요구하는 응용분야가 점차 증가함으로 인해 고성능 실시간 트랜잭션 처리 시스템 개발이 요구되고 있다. 그러나 기존에 제안된 대부분의 실시간 시스템은 다중 처리기나 분산 처리 방식을 이용하였으며, 클러스터 기술을 이용한 실시간 트랜잭션 처리 시스템은 아직 제안된 바 없다. 클러스터를 이용한 실시간 트랜잭션 처리 시스템은 저렴한 가격으로 높은 가용성과 병렬 처리를 이용한 고성능 트랜잭션 처리를 지원할 수 있다는 장점을 갖는다. 이러한 관점에서 본 논문에서는 공유 디스크(shared disks: SD) 클러스터 기반의 실시간 트랜잭션 처리 시스템을 개발하기 위하여 캐쉬 일관성 제어 기법이나 트랜잭션 라우팅 기법과 같은 전통적인 SD 클러스터 알고리즘과 실시간 트랜잭션을 위한 동시성 제어 기법을 연동한 실험 모형을 개발하였다. 다양한 환경에서의 모의실험을 통하여 알고리즘간의 상호 관계와 실시간 환경에서 SD 클러스터의 성능을 평가 분석한다.

## 1. 서 론

최근 들어, 은행, 증권시장, 기상관측, 그리고 자동화 공장의 관리 시스템과 같이 실시간 트랜잭션 처리가 필요한 응용 분야들의 규모가 커짐으로 인해 고성능 실시간 트랜잭션 처리 시스템의 필요성이 증가되고 있다. 실시간 트랜잭션은 전통적인 트랜잭션의 ACID 특징 외에도 마감기한(deadline) 내에 트랜잭션을 종료해야만 하는 시간 제약성을 가진다[8]. 실시간 트랜잭션 처리 시스템의 성능평가 기준으로는 전통적인 트랜잭션 처리 시스템에서의 시간당 트랜잭션 처리량과 함께 마감기한 내에 완료한 트랜잭션의 비율이 사용된다.

클러스터는 다수 개의 연결된 노드들이 트랜잭션 처리를 위해 하나의 노드처럼 협력하는 시스템으로, 온라인 트랜잭션 처리와 전자 상거래, 그리고 병렬 데이타베이스 시스템 등 전통적인 데이타베이스의 다양한 분야에 적용되고 있다[4]. 클러스터 구성방식은 데이타를 액세스 하는 방법에 따라 모든 노드에서 디스크를 공유하는 방식(shared disk : SD)과 메모리나 디스크를 공유하지 않는 방식(shared nothing : SN)으로 분류 할 수 있다. SD 클러스터의 경우, 모든 노드들이 디스크 계층에서 전체 데이타베이스를 공유함으로써, 다른 연동 방식들에 비해 동적 부하 분산이 용이하고 새로운 노드의 추가로 인한 확장성이 향상된다는 장점을 가진다[2].

실시간 처리와 클러스터 컴퓨팅은 각각 독립적으로 폭 넓은 연구가 이루어졌지만, 클러스터를 이용한 실시간 처리에 관한 연구는 이루어지지 않았다. 클러스터 기반의 실시간 트랜잭션 처리 시스템을 개발할 경우 다음과 같은 장점을 기대할 수 있다.

- 클러스터는 온라인 실시간 트랜잭션 처리 시스템의 높은 가용성을 지원한다.
- 클러스터는 효과적인 데이타 캐싱과 병렬처리가 가능함으로써 높은 성능의 실시간 처리가 가능하다.

이러한 관점에서 본 논문에서는 실시간 트랜잭션 처리 알고리즘과 SD 클러스터에서의 알고리즘을 적용한 SD 클러스터 기반 실시간 데이타베이스 시스템(shared disk based real-time database systems : SD-RTDBS)을 개발하기 위해 각 알고리즘들의 성능을 실험 평가한다. 본 논문에서 수행된 실험은 SD-RTDBS 모의 실험 모형을 사용하며, 캐쉬 일관성 제어 기법, 동적 트랜잭션 라우팅, 그리고 실시간 동시성 제어 기법 등을 조합하여 다양한 부하 환경에서 성능을 평가한다.

## 2. 실시간 트랜잭션 처리 알고리즘

실시간 트랜잭션 처리 시스템은 마감기한을 갖는 실시간 트랜잭션들을 효율적으로 처리하기 위해 다음과 같은 처리 과정이 필요하다. 첫째, 각 트랜잭션의 마감기한을 고려하여 우선순위를 할당한다. 이것은 스케쥴러에 의해 트랜잭션들의 처리 순서를 결정하기 위한 것으로, 효과적인 우선순위 할당정책은 마감기한 내 트랜잭션이 완료될 가능성을 높일 수 있다. 실시간 트랜잭션을 위해 제안된 기존의 우선순위 할당정책으로는 RM(rate monotonic), EDF(earliest deadline first), 그리고 LS(least slack) 기법이 있다[7]. RM 기법은 주기적인 하드 실시간 트랜잭션에만 가능한 기법으로 일반적인 실시간 트랜잭션에 대해서는 비효율적이다[3]. LS 기법은 각 트랜잭션의 실행시간을 이용하여 우선순위를 할당하는 방법으로 트랜잭션의 마감기한과 실행시간을 고려하여 우선순위를 할당함으로써 마감기한 내 트랜잭션 완료 비율이 높은 장점이 있다. 그러나 트랜잭션의 실행시간을 알 수 없기 때문에 일반적인 실시간 트랜잭션을 위해서는 적용할 수 없다[9]. 본 논문에서는 마감기한이 임박한 트랜잭션에 높은 우선순위를 할당하는 EDF 기법을 사용한다.

둘째, 동일한 데이타를 액세스하는 여러 트랜잭션들에 대한 데이타베이스의 일관성을 유지하기 위해 동시성 제어가 필요하다. 동시성 제어는 전통적인 트랜잭션 처리 시스템에서도 필요한 부분으로 크게 로크 기반의 동시성 제어기법과 낙관적인 기

법이 있다. 실시간 트랜잭션의 경우, 높은 우선순위의 트랜잭션이 낮은 우선순위의 트랜잭션에 의해 데이타를 액세스할 수 없는 우선순위 역전(priority inversion) 현상이 발생한다. 2단계 로킹 기법(2 phase locking: 2PL)은 대부분의 전통적인 트랜잭션 처리 시스템에서 동시성 제어를 위해 사용되고 있지만, 2PL만으로는 실시간 트랜잭션의 우선순위 역전 문제를 해결할 수 없다. 실시간 트랜잭션에서 우선순위 역전을 해결하기 위해 2PL 기반의 우선순위 상속(priority inheritance: 2PL-PI)과 높은 우선순위(high priority: 2PL-HP)가 제안되었다[6,7].

우선순위 역전이 발생할 경우, 2PL-HP는 우선순위가 낮은 트랜잭션을 철회함으로써 우선순위 역전 현상을 해결하며, 2PL-PI는 로크 충돌이 일어난 트랜잭션의 우선순위를 높여줌으로써 해결한다. 이 결과 2PL-HP는 잦은 트랜잭션철회로 시스템 자원의 낭비가 생기며, 2PL-PI는 우선순위 역전 현상이 지속적으로 발생하면 거의 모든 트랜잭션이 우선순위가 높아져서 문제 해결에 도움이 못될 수도 있다.

## 3. SD 클러스터를 위한 알고리즘

SD-RTDBS를 구현하기 위해서는 실시간 처리 기법들을 SD 클러스터에 맞도록 확장해야한다. 뿐만 아니라 노드들 사이의 캐쉬 일관성 기법과 부하 분산을 고려한 트랜잭션 라우팅 기법이 함께 고려되어야 한다.

### 3.1 캐쉬 일관성 관리

SD 클러스터의 각 노드는 액세스한 페이지를 자신의 버퍼에 캐싱 함으로써 참조 지역성을 이용하여 디스크 IO를 줄일 수 있다. 그러나 하나의 노드에서 페이지를 갱신할 경우 그 페이지를 캐싱하고 있는 다른 노드들에서는 무효화를 시켜야하므로 데이타베이스 페이지의 최신 버전을 유지하는 캐쉬 일관성 관리 기법(cache coherency scheme)이 필요하다.

캐시 일관성 관리 기법의 복잡성은 로크 단위에 의해 결정된다. 페이지 단위 로크의 경우, 하나의 페이지에 대해 로크 충돌이 발생하지 않으므로, 구현이 간단하다. 이에 반해 레코드 단위 로크는 동일한 페이지의 각각의 레코드들이 서로 다른 트랜잭션에 의해 갱신될 수 있기 때문에 갱신된 레코드들을 반영하기 위해 노드간의 페이지 교환 과정이 필요하다.

전통적인 데이타베이스 시스템에서 많은 트랜잭션이 동시에 실행 될 경우 페이지 단위 로크보다 레코드 단위 로크가 좀더 좋은 성능을 나타낸다. 그러나 실시간 처리 시스템에서는 동시에 실행되는 트랜잭션 수가 상대적으로 작기 때문에 레코드 단위 로크의 복잡성이 성능 저하의 원인이 될 수 있다.

### 3.2 트랜잭션 라우팅

친화도 기반 트랜잭션 라우팅은 유사한 데이타 액세스 형태를 가진 트랜잭션들을 동일한 노드에 할당함으로써 지역 버퍼의 히트율을 높이고, 노드들 간의 버퍼 무효와 로크 동기화에 의한 노드들 사이의 충돌을 줄일 수 있다[2,5]. 그러나 순수한 친화도 기반 라우팅은 부하가 정적일 때만 그 효과를 기대할 수 있으며, 동적으로 변화하는 부하의 특성을 고려한 동적 부하 분산 기법을 함께 고려해야 한다.

본 논문에서는 다음 세 가지 라우팅 기법들을 고려한다.

- 정적 트랜잭션 라우팅(pure affinity based routing : PAR)은 트랜잭션 처리 시간을 줄일 수 있지만 시스템의 부하가 변하면 효율적이지 못하다[5].

- 라운드-로빈 방식의 동적 트랜잭션 라우팅(dynamic round-robin routing : DRR)은 시스템의 부하가 균등히 분산될 때 친화도 기반의 트랜잭션 라우팅을 수행하고, 부하가 변할 때 폭주하는 클래스의 트랜잭션을 라운드-로빈 방식으로 모든 노드에 걸쳐서 분산시킨다. DRR은 동적 부하 분산으로 시스템의 부하 균형을 유지할 수 있지만, 낮은 버퍼 히트율과 노드들 사이에 잦은 버퍼 무효화로 인해 성능 향상에 한계가 있다[2].

- 친화도 클러스터의 동적 할당(dynamic affinity cluster allocation : DACA)은 동적 부하 분산과 친화도 기반 라우팅을 최적화 할 수 있다[2]. DACA는 다음과 같은 두 가지 특징을 가진다. 첫째, 각 노드의 부하 편차가 급격하지 않는 범위에서 트랜잭션 클래스에 할당된 노드의 수를 최소로 유지함으로써 버퍼 무효화를 줄일 수 있다. 둘째, 과부하 된 트랜잭션 클래스와 다른 트랜잭션 클래스들을 동일한 노드에서 실행시키지 않음으로써 높은 버퍼 히트율을 유지한다.

## 4. 실험방법

본 논문에서는 SD-RTDBS에서 일관성 관리, 동시성 제어 기법, 그리고 라우팅 기법들을 평가하기 위한 모의실험 모형을 개발하였다. 표 1은 실시간 트랜잭션의 마감기한에 사용되는 매개변수들을 설명한다. 각 트랜잭션의 마감기한은 식 (1)을 이용하여 계산하며 트랜잭션 실행시간 $E_T$는 식 (2)와 같이 계산된다. 본 논문에서는 *ARIES/SD*[1]의 *steal*과 *not force* 버퍼 관리 정책을 구현하였기 때문에 갱신된 페이지 연산에서 디스크 반영 시간을 포함하지 않는다.

표 1 마감기한 매개 변수

| 매개 변수 | 설 명 |
|---|---|
| $D_T$ | 트랜잭션 T의 마감기한 |
| $A_T$ | 트랜잭션 T의 도착 시간 |
| $E_T$ | 트랜잭션 T의 실행 시간 |
| $SF$ | 마감기한의 긴박성 정도 |
| $NumReads_T$ | 트랜잭션 T가 판독하는 페이지 수 |
| $NumWrites_T$ | 트랜잭션 T가 갱신하는 페이지 수 |
| $PageCpu$ | 페이지의 CPU 처리 시간 |
| $PageDisk$ | 페이지의 디스크 시간 |

$$D_T = A_T + SF \times E_T \quad (1)$$

$$E_T = NumReads_T \times (PageCpu + PageDisk) + NumWrites_T \times PageCpu \quad (2)$$

실험에서 사용한 주요 성능 평가 지수는 트랜잭션 처리율과 마감기한 위반율이다. 트랜잭션 처리율은 단위 시간동안 완료된 트랜잭션의 수에 의해 측정된다. 마감기한 위반율은 전체 실행된 트랜잭션 수에서 교착상태에 의한 철회 트랜잭션을 제외한 트랜잭션 중에 마감기한을 위반한 트랜잭션 수의 비율이다. 그리고 보조 성능 지수로 트랜잭션을 실행하기 위해 요청된 페이지를 지역노드 혹은 다른 노드의 버퍼에서 판독할 확률을 나타내는 버퍼 히트율을 사용한다.

표 2 알고리즘 비교

| 기 법 | 비교 알고리즘 |
|---|---|
| 로크 단위 | 페이지 단위, 레코드 단위 |
| 동시성 제어 | 2PL-HP, 2PL-PI |
| 트랜잭션 라우팅 | PAR, DRR, DACA |

표 2는 본 논문에서 평가할 기법들을 나타낸다. 로크 단위, 동시성 제어 기법, 그리고 트랜잭션 라우팅의 3가지 기법을 비교 한다. 각 기법들에 대하여, 총 12가지의 SD-RTDBS 알고리즘 조합이 가능하다.

## 5. 실험 결과

그림 1은 동시에 실행되는 트랜잭션의 수(MPL)의 변화에 따른 페이지 단위 로킹과 레코드 단위 로킹의 성능이다. MPL이 증가함에 따라 로크 단위에 관계없이 성능이 감소한다. 그 이유는 트랜잭션들 사이의 자원 경쟁이 심각해지기 때문이다. 단위 시간당 트랜잭션 처리량의 경우 레코드 단위 로킹이 높은 동시성으로 인해 우수한 성능을 보인다. 그러나 마감기한 위반율의 경우, 레코드 단위 로킹의 복잡성으로 인해 높은 동시성에도 불구하고 비슷한 결과를 보인다.

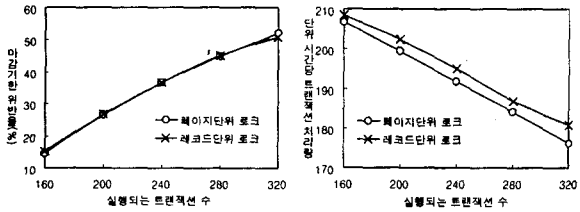


그림 1 2PL-HP, DACA 기반에서 로크 단위의 성능 평가

그림 2는 특정 트랜잭션 클래스의 부하가 변할 때, 트랜잭션 처리율과 마감기한 위반율의 모의 실험 결과이다. 라우팅 기법의 경우, 부하 폭주율이 0%일 때 모든 노드의 부하가 동일하기 때문에 DACA나 DRR과 같은 동적 라우팅 기법들도 PAR과 동일한 방식을 수행한다. DACA는 20%에서 80%의 부하 폭주율 사이에서 높은 버퍼 히트율로 인해 성능이 우수하다. 로크 단위의 경우, 높은 동시성을 가지는 레코드 단위 로킹이 페이지 단위 로킹보다 우수한 성능을 나타낸다.

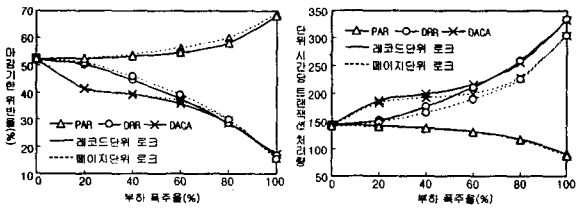


그림 2 2PL-HP 기반의 로크 단위 및 트랜잭션 라우팅 알고리즘 비교

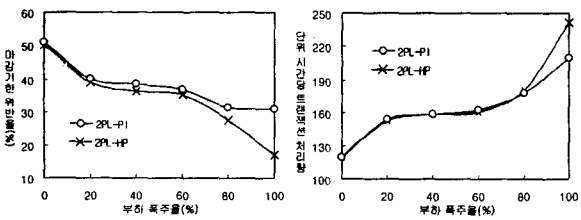


그림 3 DACA 기반의 실시간 동시성 관리 알고리즘 비교

그림 3에서는 특정 트랜잭션 클래스의 부하를 변화시킬 때 동시성 관리 기법들 간의 트랜잭션 처리율과 마감기한 위반율이 나타난다. 부하가 0%에서 80%까지 단위 시간당 트랜잭션 처리량은 그림 3의 (b)와 같이 두 기법이 비슷하다. 그러나 그림 3의 (a)에서 나타나듯이 마감기한 위반율은 2PL-PI가 높다. 그 이유는 우선순위 역전이 발생했을 때, 낮은 우선순위의 트랜잭션을 철회시키는 것이 아니라 우선순위를 높여서 처리하기 때문에 많은 트랜잭션들 높은 우선순위를 가지게 되고, 결국 높은 우선순위의 트랜잭션들 사이에 데이타 충돌이 발생하기 때문이다.

## 6. 결론 및 향후 과제

고성능 실시간 트랜잭션 처리를 위하여 여러 대의 컴퓨터를 연동하는 클러스터는 처리 능력과 가용성, 그리고 비용의 측면에서 장점을 가진다. 본 논문에서는 SD 클러스터를 실시간 처리 시스템에 적용하고, 기존에 기법들의 성능을 비교 하였다.

SD-RTDBS의 성능을 분석하기 위하여 모의실험 환경을 구축하였고, 다양한 데이타베이스 부하와 시스템 구성 하에서 실험을 수행하였다. 실험의 결과로 DACA는 지역 버퍼에 대한 참조 지역성을 높이고 버퍼 무효화 오버헤드를 줄임으로써 시스템의 성능을 향상시켜 부하가 동적으로 변화할 때 뛰어난 성능을 보였고, 레코드 단위 로킹이 높은 동시성으로 인하여 페이지 단위 로킹보다 좋은 성능을 보였다. 동시성 제어 기법의 경우 성능 차이가 많이 나지는 않았으나 2PL-HP가 2PL-PI보다 좀 더 나은 성능을 보였다. 그 이유는 2PL-PI가 시스템 자원을 낭비 하지 않아 단위 시간당 트랜잭션 처리율은 좋지만, 트랜잭션의 여유 시간이 짧은 트랜잭션을 기다리게 함으로써 트랜잭션의 마감기한을 맞추기가 어렵기 때문이다.

단위 시간당 트랜잭션 처리율이 높은 기법일수록 시스템 자원을 효율적으로 사용하여 실행기한 위반율을 감소시킬 수 있었다. 이것으로 인해 실험 결과 전통적인 트랜잭션 처리 시스템에서 성능이 좋은 기법들이 실시간 트랜잭션 처리시스템에서도 좋게 나타났다.

본 논문의 향후 과제로는 SD 클러스터 기반의 실시간 트랜잭션 처리 시스템을 개발하고, 제안된 알고리즘들의 성능을 평가하는 것이다.

## 참고문헌

[1] C. Mohan and I. Narang, "Recovery and Coherency Control Protocols for Fast Intersystem Page Transfer and Fine-Granularity Locking in a Shared Disks Transaction Environment," *Proc. 17th VLDB Conf.*, pp. 193-207, 1991.

[2] K. Ohn and H. Cho, "Cache Conscious Dynamic Transaction Routing in a Shared Disks Cluster," *Lecture Notes in Computer Science* (2004) acceptd for publication.

[3] K. Lam, T. Kuo, *Real-Time Database Systems*, Kluwer Academic Publishers, 2001.

[4] M. Yousif, "Shared-Storage Clusters," *Cluster Comp.*, Vol.2, No.4 pp 249-257, 1999.

[5] P. Yu and A. Dan, "Performance Evaluation of Transaction Processing Coupling Architectures for Handling System Dynamics," *IEEE Trans. on Parallel and Distributed Syst.*, Vol.5, No.2, pp.139-153, 1994.

[6] R. Abbott and H. Garcia-Molina, "Scheduling Real-Time Transactions with Disk Resident Data," *Proc of 15th Intl. Conf. on VLDB*, 1989.

[7] R. Abbott and H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation," *ACM Trans. Database Systems*, Vol. 17, No. 3, 1992.

[8] S. Takkar and S. Dandamudi, "Performance of Hard Real-Time Transaction Scheduling Policiesin Parallel Database Systems," *Proc. Int. Symp. Computer and Telecommunication Syst.*, pp.176-184, 1998.

[9] V. Kanitkar and A. Delis, "Real-time Processing in Client-Server Databases," *IEEE Trans. on Computers*, Vol. 51, No. 3, 2002.