공유 디스크 클러스터에서 실시간 트랜잭션 처리의 성능 평가

(Performance Evaluation of Real-Time Transaction Processing in a Shared Disk Cluster)

이 상 호 † 온 경 오 †† 조 행 래 †††
(Sangho Lee) (Kyungoh Ohn) (Haengrae Cho)

요 약 공유 디스크(Shared Disks: SD) 클러스터는 다수 개의 처리 노드를 연동하는 방식으로, 각 노드는 디스크 계층에서 데이타베이스를 공유한다. 고성능의 트랜잭션 처리를 위한 SD 클러스터의 효율성은 기존의 연구들을 통해서 입증되었으나, SD 클러스터 기반의 실시간 처리에 대한 연구는 지금까지 이루어지지 않았다. 실시간 트랜잭션의 경우 전통적인 트랜잭션의 ACID 속성 외에 시간 제약성을 추가로 가진다. 클러스터 기술을 실시간 트랜잭션 처리에 도입함으로써 높은 가용성과 노드들 사이의 병렬성에 따른 성능 향상을 기대할 수 있다. 이런 관점에서 본 논문에서는 먼저 SD 클러스터 기반 실시간 데이타베이스 시스템의 실험 모형을 개발한다. 그리고 개발한 모형을 기반으로 실시간 트랜잭션 처리를 위한 SD 클러스터의 적합성 여부를 평가한다. 뿐만 아니라 실시간 트랜잭션 처리 알고리즘과 SD 클러스터 알고리즘들 간의 상호 영향에 대해 다양한 실험을 통하여 평가한다.

키워드 : 성능평가, 클러스터 컴퓨팅, 실시간 트랜잭션 처리

Abstract A shared disks (SD) cluster couples multiple computing nodes, and every node shares a common database at the disk level. A great deal of research indicates that the SD cluster is suitable to high performance transaction processing, but the aggregation of SD cluster with real-time processing has not been investigated at all. A real-time transaction has not only ACID properties of traditional transactions but also time constraints. By adopting cluster technology, the real-time services will be highly available and can exploit inter-node parallelism. In this paper, we first develop an experiment model of an SD-based real-time database system (SD-RTDBS). Then we investigate the feasibility of real-time transaction processing in the SD cluster using the experiment model. We also evaluate the cross effect of real-time transaction processing algorithms and SD cluster algorithms under a wide variety of database workloads.

Key words : performance evaluation, cluster computing, real-time transaction processing

1. 서 론

통신 시스템이나 증권 시장, 그리고 전자 상거래와 같이 실시간 트랜잭션 처리가 필요한 응용 분야들의 규모가 커지고 있다. 실시간 트랜잭션은 전통적인 트랜잭션의 ACID 속성 외에도 마감기한(deadline) 내에 트랜잭션을 완료해야 하는 시간 제약성을 가진다[1,2]. 실시간 트랜잭션 처리 시스템에서의 중요한 성능평가 기준으로는 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 등의 다양한 분야에 적용되고 있다. 클러스터의 구성방식은 데이타를 액세스하는 방법에 따라 모든 노드에서 디스크를 공유하는 방식(Shared Disks: SD)과 메모리나 디스크를 공유하지 않는 방식(Shared Nothing: SN)으로 분류할 수 있다[3]. SN 클러스터의 각 노드들이 할당된 데이타베이스 분할에 대해서만 직접 액세스가 가능한데 반하여, SD 클러스터는 디스크

· 이 논문은 2004학년도 영남대학교 학술연구조성비지원에 의한 것임
† 비 회 원 : 영남대학교 컴퓨터공학과
comman35@yumail.ac.kr
†† 정 회 원 : 영남대학교 정보통신 연구소 교수
ondal@yumail.ac.kr
††† 종신회원 : 영남대학교 컴퓨터공학과 교수
hrcho@yu.ac.kr
논문접수 : 2004년 7월 19일
심사완료 : 2004년 12월 28일

계층에서 전체 데이타베이스를 공유함으로써 동적 부하 분산이 용이하고 새로운 노드의 추가로 인한 확장성이 향상된다는 장점을 가진다. 뿐만 아니라, SAN(storage area networks)이나 NAS(network attached storage) 기술의 발전으로 높은 가용성과 데이타 액세스의 융통성을 지원하는 SD 클러스터의 장점이 더욱 더 부각되고 있다. 이러한 SD 클러스터를 이용한 병렬 데이타베이스 시스템으로는 IBM DB2 Parallel Edition[4]과 Oracle Real Application Cluster[5] 등이 있다.

실시간 처리와 SD 클러스터는 각각 독립적으로 폭넓은 연구가 이루어졌지만, SD 클러스터를 이용한 실시간 처리에 관한 연구는 이루어지지 않았다. SD 클러스터를 이용할 경우, 통신 서비스의 핵심인 실시간 데이타베이스 서비스의 높은 가용성을 보장할 수 있다. 뿐만 아니라, 노드들 사이의 병렬성과 데이타의 효율적인 캐싱은 디스크 IO를 감소시켜 고성능 실시간 트랜잭션 처리가 가능하다. 그러나 SD 클러스터의 경우 캐쉬 일관성 정책에 따른 추가적인 노드간 동기화 오버헤드가 존재하므로 이 오버헤드가 실시간 트랜잭션 처리에 미치는 영향에 대한 분석이 필요하다. 특히 실시간 SD 클러스터에서 요구되는 실시간 트랜잭션 라우팅 정책이나 실시간 캐쉬 일관성 정책이 전무한 상태에서 기존 SD 클러스터 알고리즘을 실시간 환경에 적용하여 성능 특성을 파악하는 것은 효율적인 실시간 SD 클러스터 알고리즘을 개발하기 위해 필요한 과정이라고 판단된다.

이런 관점에서 본 논문에서는 SD 클러스터 기반의 실시간 데이타베이스 시스템(SD cluster based real-time database system: SD-RTDBS) 실험 모형을 개발하여 실시간 트랜잭션 처리 시 실시간 처리와 SD 클러스터 알고리즘들이 각각에 미치는 영향을 분석한다. 본 논문에서 실시간 처리와 SD 클러스터 알고리즘들 간의 관계에서 중요하게 고려한 사항들은 다음과 같다.

- 실시간 트랜잭션 처리에 SD 클러스터가 적합한가?
- SD 클러스터에서의 실시간 트랜잭션 실행결과는 전통적인 트랜잭션의 실행결과와 얼마나 다르게 나타나는가?
- SD 클러스터가 실시간 트랜잭션 처리 알고리즘의 성능에 어떤 영향을 미치는가? 특히, 실시간 트랜잭션을 위한 동시성 제어 알고리즘의 성능에 어떤 영향을 미치는가?

본 논문의 구성은 다음과 같다. 2절에서는 본 논문에서 고려한 SD-RTDBS에서의 실시간 처리 알고리즘에 대해 설명한다. 3절에서는 SD-RTDBS의 실험 모형에 대해서 설명하고 4절에서는 성능 평가 결과를 분석한다. 마지막으로 5장에서 결론을 맺는다.

2. 실시간 트랜잭션 처리

본 논문에서는 SD-RTDBS에서 고려해야 할 실시간 트랜잭션 처리 기법들을 캐쉬 일관성 관리 기법, 트랜잭션 라우팅 기법, 그리고 동시성 제어 기법으로 분류하였다. 이중 캐쉬 일관성 관리 기법과 트랜잭션 관리 기법은 실시간 알고리즘이 존재하지 않으므로 기존 SD 클러스터에서 제안된 알고리즘들을 중심으로 고려하였고, 동시성 제어 기법은 SD 클러스터가 아닌 기존 실시간 시스템에서 제안된 실시간 알고리즘들을 고려하였다. 본 절에서는 각 기법별로 기존에 제안된 알고리즘들을 설명하고 각 알고리즘들에 대한 성능 평가의 필요성을 설명한다.

2.1 캐쉬 일관성 관리 기법

SD 클러스터에서 각각의 노드는 자신의 버퍼에 최근에 액세스한 데이타들을 캐싱한다. 데이타에 대한 효율적인 캐싱은 각 노드에서 발생하는 디스크 액세스 수나 노드들 간의 데이타 전송량을 줄임으로써 SD 클러스터의 성능을 크게 향상시킬 수 있다. 그러나 노드들이 최신의 데이타를 항상 사용할 수 있기 위해서는 버퍼에 캐싱된 데이타의 일관성이 유지되어야 한다. 이를 위해 각 노드에 존재하는 버퍼 관리자는 캐쉬 일관성 관리 기법을 지원하여야 하는데, 기본 개념은 한 노드에서 특정 데이타를 갱신할 경우 그 데이타의 이전 버전을 캐싱하고 있는 다른 노드들의 버퍼에서 무효화를 시킨다는 것이다[6-8].

캐쉬 일관성 관리 기법의 복잡성은 로크의 단위에 의해 결정된다. 페이지 단위 로킹의 경우, 서로 다른 트랜잭션들에 의해 로크 충돌이 발생하는 페이지에 대한 액세스를 허용하지 않기 때문에 캐쉬 일관성 관리가 레코드 단위 로킹보다 간단하다. 레코드 단위 로킹의 경우, 캐쉬 일관성 관리가 매우 복잡하다. 그 이유는 서로 다른 노드가 동일한 페이지에 포함된 상이한 레코드에 대해서 동시에 갱신이 가능하기 때문이다. 이 경우 각 노드의 버퍼에 캐싱된 페이지는 부분적으로 최신의 데이타를 유지하며, 갱신된 내용이 디스크에는 포함되지 않을 수도 있다. 결국, 갱신된 레코드들을 모두 페이지에 반영하기 위해서는 노드간의 페이지 교환을 위한 추가적인 메시지 전송이 발생한다.

2.2 트랜잭션 라우팅 기법

SD-RTDBS에서 주요 고려 사항은 노드들 간의 동기화를 위한 오버헤드를 최소로 줄이면서 노드들 간의 병렬성을 극대화하는 것이다. 이를 위해 SD-RTDBS는 효율적인 트랜잭션 라우팅 기법을 수행해야 한다. SD 클러스터에서 친화도 기반 라우팅(affinity-based routing)은 유사한 데이타 액세스 형태를 가진 트랜잭션들을

동일한 노드(affinity node)에 할당함으로써 지역 버퍼의 히트율을 높이고, 노드들 간의 버퍼 무효화와 로크 동기화에 의한 노드들 사이의 충돌을 줄일 수 있다[9-11].

본 논문에서는 시스템의 동적인 부하의 변화를 고려하지 않는 친화도 기반 트랜잭션 라우팅(pure affinity-based routing: PAR)[10,11]과 동적인 부하 분산을 고려하는 친화도 기반 동적 트랜잭션 라우팅(dynamic affinity-based cluster allocation: DACA)[9]을 고려한다. PAR는 트랜잭션 클래스와 노드들 사이의 친화도 관계가 고정되어 있는 정적인 기법으로, 구현이 쉽고 간단하다는 장점을 가진다. 하지만 특정 트랜잭션 클래스의 부하가 증가할 경우 해당 친화도 노드가 과부하 상태가 되어 트랜잭션 처리율이 감소한다는 단점을 갖는다. DACA는 SD 클러스터에서 동적인 부하 분산과 친화도 기반 트랜잭션 라우팅을 최적화하기 위하여 제안된 알고리즘이다. 이를 위하여 DACA는 기본적으로 친화도 기반의 트랜잭션 라우팅을 수행하면서 시스템의 부하가 변할 경우 동적으로 친화도 관계를 변경함으로써 동적인 부하 분산을 수행한다. 즉, 특정 트랜잭션 클래스의 부하가 폭주할 때 노드를 추가 할당하여 부하를 적정 수준으로 유지하고, 노드들 간의 부하 편차가 크지 않을 경우 폭주 트랜잭션 클래스에 추가로 할당된 친화도 노드들의 수를 제한하여 버퍼 무효화를 줄인다.

2.3 동시성 제어 기법

동시에 실행되는 트랜잭션들에 대한 데이타베이스 일관성을 유지하기 위하여 동시성 제어 기법이 필요하다. 실시간 환경에서의 동시성 제어 기법은 높은 우선순위의 트랜잭션이 낮은 우선순위의 트랜잭션에 의해서 실행이 지연되는 우선순위 역전(priority inversion)을 해결해야 한다. 우선순위 역전이 발생할 경우 낮은 우선순위를 가진 트랜잭션이 완료할 때까지 마감기한이 임박한 높은 우선순위의 트랜잭션이 실행되지 못하기 때문에 마감기한 위반율이 증가한다.

대표적인 실시간 동시성 제어 기법으로 2단계 로킹(2PL) 기반의 WP((wait promote)와 HP(high priority)를 들 수 있다[12]. WP는 우선순위 역전이 발생할 때 우선순위 역전의 원인이 되는 트랜잭션의 우선순위를 대기 중인 트랜잭션의 우선순위만큼 증가시킴으로써 우선순위 역전을 해결한다. HP는 우선순위 역전현상의 원인이 되는 트랜잭션을 철회시킴으로써 문제를 해결한다. WP와 HP는 1990년대 초반에 제안된 알고리즘들이지만 2PL을 실시간 환경으로 확장하기 위한 기본적인 개념을 지원한다는 관점에서 오늘날까지 널리 채택되어 사용되고 있다[13-15]. 실시간 동시성 제어를 위한 낙관적인 알고리즘[16]이나 직렬화 그래프 검사 알고리즘[17] 등도 제안된 적이 있으나, 대부분의 상용 데이타베이스 시스템들이 2PL을 구현하고 있으므로 본 논문에서도 2PL 기반 알고리즘인 WP와 HP를 주로 고려하도록 한다.

2.4 성능 평가의 필요성

본 논문에서는 앞에서 설명한 세 가지 분야에 대해서 SD-RTDBS 실험 모형을 구현하여 SD 클러스터 상에서 실시간 트랜잭션 처리의 성능을 비교한다. 각 분야에 대해 성능 평가의 대상이 되는 알고리즘들은 다음과 같다.

- 캐쉬 일관성 기법 - 레코드 로킹 기반, 페이지 로킹 기반
- 트랜잭션 라우팅 기법 - DACA, PAR
- 실시간 동시성 제어 기법 - HP, WP

각 알고리즘들 간의 성능 평가는 각각 다양한 환경에서 수행된 바 있으나, 본 논문과 같이 SD 클러스터 상에서 실시간 트랜잭션 처리의 관점에서는 성능 평가가 수행된 바 없다. 본 절에서는 기존의 실험 결과들을 간단히 소개하고, 실시간 SD 클러스터 환경에서 성능 평가가 추가로 필요한 이유를 정리한다.

먼저 캐쉬 일관성 기법의 경우, 레코드 단위 로킹은 로크 충돌 비율을 감소시킬 수 있다. 따라서 많은 트랜잭션들이 동시에 실행되는 전통적인 트랜잭션 처리 시스템에 적합하다. 이러한 이유로 대부분의 상용 SD 클러스터들은 레코드 단위 로킹을 지원한다[4,5]. 그러나 레코드 단위 로킹은 로크 요청 및 로크 허용을 위한 메시지의 수가 증가하고 캐쉬 일관성 유지를 위한 추가적인 메시지 전송이 필요하다. 뿐만 아니라 과도한 페이지 전송은 로그 레코드 기록으로 인한 빈번한 디스크 IO의 원인이 된다[8]. 레코드 단위 로킹의 이러한 단점은 동시성 정도가 높지 않은 실시간 환경에서 특히 문제가 된다. 따라서 본 논문에서는 다양한 실시간 부하환경에서 로크 단위가 캐쉬 일관성 관리 기법에 미치는 영향에 대해 성능 평가를 수행하고자 한다.

다음으로 트랜잭션 라우팅 기법의 경우, DACA가 PAR를 포함한 다른 동적 트랜잭션 라우팅 알고리즘들보다 전통적인 트랜잭션 처리에서 좋은 성능을 보였다[9]. 그러나 주의할 점은 전통적인 트랜잭션 처리 환경의 주요 성능 지수는 단위 시간당 트랜잭션 처리 수나 평균 트랜잭션 응답시간이라는 점이다. 이와는 달리 실시간 트랜잭션 처리 환경의 주요 성능 지수는 마감기간 위반율이다. 즉, 실시간 트랜잭션 처리 환경에서는 단위 시간당 많은 수의 트랜잭션을 처리할 수 있는 알고리즘보다는 마감기간이 얼마 남지 않은 트랜잭션을 우선적으로 처리할 수 있는 알고리즘이 요구된다. 이런 관점에서 본 논문에서는 실시간 트랜잭션들을 위한 동시성 제어 기법이 적용된 환경에서 마감기한 위반율에 대한

PAR와 DACA의 성능을 비교한다.

마지막으로 실시간 동시성 제어 기법의 경우, 중앙 집중형 데이타베이스 시스템에서의 성능 평가는 다음과 같이 나타났다. 먼저, 마감기한을 넘긴 트랜잭션의 실행 결과는 의미가 없는 것으로 간주되는 경성 실시간(firm real-time) 트랜잭션에서는 HP가 WP보다 좋은 성능을 보이는 것으로 보고 되었다[16]. 그 이유는 HP의 경우 마감기한을 넘긴 트랜잭션의 실행 확률이 WP보다 작기 때문이다. 그러나 마감기한을 넘긴 트랜잭션의 실행 결과도 가치가 있는 연성 실시간(soft real-time) 트랜잭션의 경우에는 HP의 높은 트랜잭션 철회율이 성능에 부정적인 영향을 미치고, 그 결과 WP가 HP보다 좋은 성능을 나타내었다[12]. 실시간 SD 클러스터의 경우, 다음과 같은 이유로 실시간 트랜잭션을 위한 동시성 제어 기법들의 성능 평가가 필요하다. 첫째, SD 클러스터의 각 노드에서 동시에 실행되는 트랜잭션 수가 증가하고 따라서 데이타 충돌 비율이 중앙 집중형 데이타베이스 시스템에 비해서 증가한다. 둘째, 캐쉬 일관성 기법과 SD 클러스터를 위한 트랜잭션 라우팅 기법이 동시에 수행될 경우 동시성 제어 기법에 영향을 미칠 수 있기 때문이다.

3. 실험 모형

본 논문에서는 이산-사건 실험 프로그램인 CSIM 언어[18]를 이용하여 SD-RTDBS의 성능 평가 모형을 구현하였다. 그림 1은 구현한 실험 모형이다.

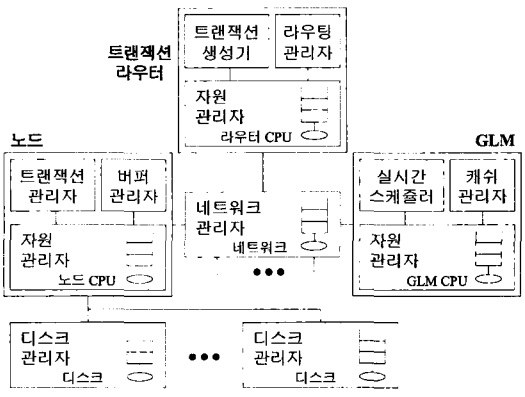


그림 1 SD 클러스터의 성능 평가 모형

SD 클러스터는 하나의 라우터와 전역 로크 관리자(global lock manager: GLM), 그리고 네트워크로 연결된 여러 개의 노드로 구성되고, 각 노드는 모든 디스크들을 공유한다. 라우터는 트랜잭션 생성기와 라우팅 관리자로 구성된다. 트랜잭션 생성기는 데이타베이스 연산들로 구성된 트랜잭션을 생성하는 역할을 담당하고, 라우팅 관리자는 PAR와 DACA의 라우팅 기법을 구현한다.

각 노드는 LRU 버퍼를 관리하는 버퍼 관리자, 그리고 CPU와 공유 디스크, 네트워크 등의 자원을 관리하는 자원 관리자를 가진다. 트랜잭션 관리자는 각 트랜잭션에 대해 GLM에 로크 요청 메시지와 트랜잭션 완료 메시지를 전송한다.

GLM은 전역 실시간 동시성 제어와 캐쉬 일관성 관리를 수행하는 역할을 한다. 실시간 스케쥴러는 WP 또는 HP을 수행하며, 로크 단위는 레코드 또는 페이지로 정의된다. 트랜잭션은 로크 충돌로 인하여 대기상태가 되고, 교착상태 또는 HP 기법에서 우선순위 역전이 발생할 경우 철회될 수 있다. 캐쉬 관리자는 SD 클러스터의 대표적인 캐쉬 일관성 기법인 ARIES/SD 기법을 구현하였다[8].

실험에 사용된 시스템 구성 변수와 오버헤드를 표 1에서 요약한다. 각 매개 변수의 구체적인 값들은[7,11]에서 주로 참조하였다. 네트워크 관리자는 100 Mbps의 대역폭을 갖는 FIFO서버로 구현하였다. 공유 디스크 수는 20개이며 각 디스크는 I/O 요청을 우선순위에 따라 처리한다. 디스크 액세스 시간은 0.01초와 0.03초 사이의 일양 분포를 따른다고 가정한다. 네트워크를 통해 메시지를 전송하는 과정을 표현하기 위해 각 노드와 GLM의 CPU는 메시지마다 MsgInst만큼의 고정된 명령수를 실행하고, 제어 메시지의 경우에는 CtlMsgSize / NetBandwidth 그리고 페이지를 포함한 메시지의 경우에는(PageSize + CtlMsgSize) / NetBandwidth 만큼의 네트워크 지연을 갖는다.

데이타베이스는 여러 개의 논리적인 클러스터로 구성되며, 하나의 데이타베이스 클러스터는 10000개의 페이지(40 Mbytes)를 가진다. 하나의 트랜잭션 클래스는 하나의 데이타베이스 클러스터와 연관되며, 트랜잭션 클래스의 수는 8로 설정하였다. Locality는 트랜잭션 클래스가 연관된 데이타베이스 클러스터에서 참조하는 데이타의 비율을 의미한다. 각 데이타베이스 클러스터는 20%의 hot set(HotSize)을 가지고 트랜잭션들의 80%(HotPr)가 이를 액세스 하는 "80-20 규칙"을 따른다. 트랜잭션에 의해 액세스되는 평균 레코드 수는 TrxSize ± TrxSize × SizeDev 사이의 일양 분포를 따른다. UpdatePr는 갱신되는 레코드의 비율을 의미한다. 각 레코드를 처리하는 데는 15000개의 명령어(PerObjInst)가 실행된다고 가정한다.

트랜잭션 T의 마감기한 DT는 식 (1)에 의해 계산된다. A_T와 E_T는 T의 도착 시간과 실행시간을 의미한다. SF(slack factor)는 트랜잭션의 마감기한의 긴박 정도를

표 1 입력 매개 변수

시스템 구성 변수		
CPUSpeed	노드 CPU의 속도	1 GIPS
NetBandwidth	네트워크의 데이타 전송 속도	100 Mbps
NumNode	노드의 수	1 ~ 16
NumDisk	공유 디스크의 수	20
DiskTime	디스크 액세스 시간	0.01 ~ 0.03 초
PageSize	페이지 크기	4096 bytes
RecPerPage	페이지당 레코드 수	10
ClusterSize	클러스터의 페이지 수	10000
HotSize	클러스터의 hot set의 페이지 수	2000
DBSize	데이타베이스의 클러스터 수	8
BufSize	노드의 버퍼 크기(페이지 수)	4000
CtlMsgSize	제어 메시지 길이	256 bytes
MsgInst	메시지당 명령 수	22000
LockInst	로크 획득/해제를 위한 명령 수	2000
PerIOInst	디스크 I/O를 위한 명령 수	5000
PerObjInst	레코드 처리를 위한 명령 수	15000
LogIOTime	로그 레코드 기록 시간	0.005 초
트랜잭션 변수		
TrxSize	트랜잭션이 액세스하는 레코드 수	10
SizeDev	트랜잭션 크기 편차	0.2
WriteOpPr	레코드 갱신 비율	0.2 ~ 0.08
MPL	동시에 실행되는 트랜잭션 수	40 ~ 400
ClassNum	트랜잭션 클래스의 수	8
Locality	지역 클러스터를 액세스할 확률	0.8
HotPr	hot set을 액세스할 확률	0.8

의미하며, 본 논문에서는 [16]에서와 같이 SF를 4.0으로 설정한다. ET는 식 (2)로 계산되며, $NumReads_T$와 $NumWrites_T$는 트랜잭션 T의 읽기 연산과 갱신 연산의 수이다. 식 E_T에서 갱신된 페이지 쓰기의 디스크 시간이 포함되지 않은 이유는 트랜잭션이 완료한 후에 쓰기 연산을 수행하기 때문이다[16].

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

$$E_T = NumReads_T \times (DiskTime + PerObjInst) + NumWrites_T \times PerObjInst \quad (2)$$

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

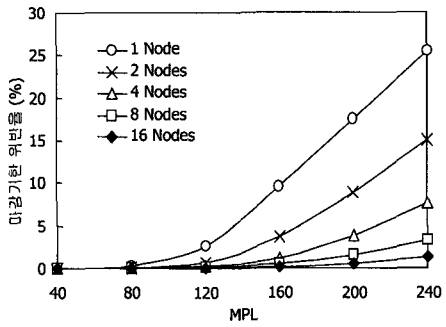
4. 실험 결과

본 절에서는 2절에서 설명한 세 가지 분야에 대해서 다음과 같이 성능을 비교한다. 첫째, 트랜잭션 라우팅 기법으로 PAR와 DACA를 비교한다. 둘째, 로크의 단위를 페이지와 레코드로 설정하여 캐쉬 일관성 관리 기법의 복잡성을 비교한다. 마지막으로 동시성 제어 기법인 WP와 HP를 비교한다. 본 논문에서는 분류된 기법들을 조합하여 총 8가지의 SD-RTDBS 기법들을 구현하였다.

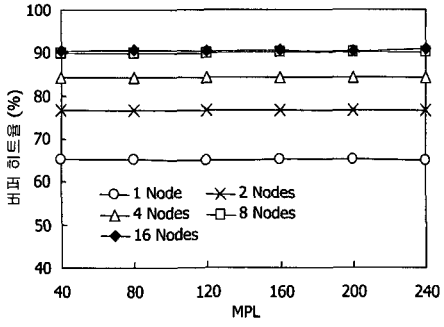
4.1 실험 1 : SD 클러스터를 이용한 실시간 처리의 효율성

SD 클러스터에서 실시간 처리의 효율성을 검증하기 위하여 노드의 수(NumNode)와 동시에 실행되는 트랜잭션의 수(MPL)를 변화시키면서 성능을 비교한다. 그림 2의 (a)는 트랜잭션 클래스의 수(ClassNum)를 1로 설정하고 실험한 결과이다. 로크의 단위는 레코드이며 실시간 동시성 제어를 위해 HP를 수행하였다. 그리고 라우팅 기법은 DACA를 수행하였다. 갱신되는 레코드의 비율은 0.2로 설정하였다.

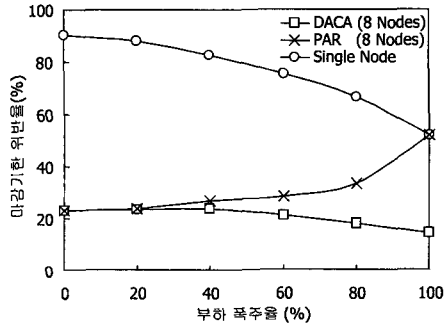
MPL이 낮은 경우, 단일 노드에서도 마감기한 안에 트랜잭션을 완료할 가능성이 높기 때문에 모든 경우에서 성능이 비슷하다. 그러나 MPL이 증가함에 따라 부하 분산과 높은 버퍼 히트율로 인해 노드 수가 많은 시스템의 성능개선 효과가 명확하게 나타난다. 노드 수가 많을 경우 시스템의 부하가 분산되고, 전체 버퍼 크기가 증가되어 그림 2의 (b)와 같이 버퍼 히트율이 증가한다. 결국, 다수 개의 노드를 연동할 경우, 각 노드들 간의 동기화로 인한 오버헤드가 있음에도 불구하고 부하 분산과 높은 버퍼 히트율에 의한 성능 개선이 더 크게 나



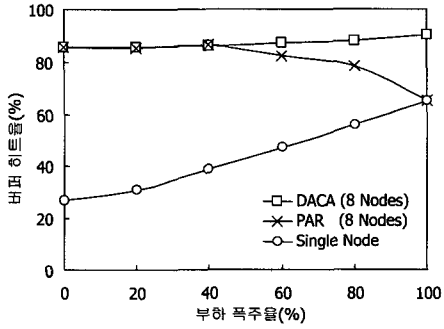
(a) MPL 변화에 따른 마감기한 위반율



(b) MPL 변화에 따른 버퍼 히트율



(c) 동적 부하 변화에 따른 마감기한 위반율



(d) 동적 부하 변화에 따른 버퍼 히트율

그림 2 실험 결과 1

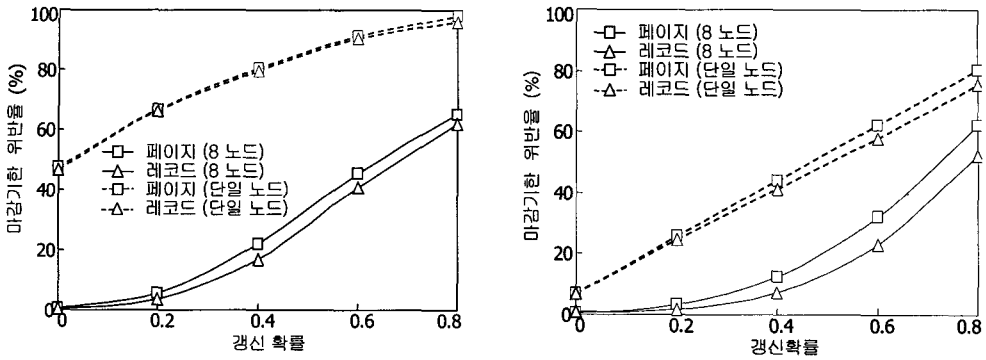
타났다.

그림 2의 (c)와 (d)는 다수 개의 노드를 연동하면서 트랜잭션 라우팅을 위해 PAR와 DACA를 수행할 때와 단일 노드 시스템의 성능을 비교한 것이다. NumNode와 ClassNum 모두 8개로 설정하였고 MPL은 400으로 설정하였다. 따라서 부하 균형 상태에서 각 트랜잭션 클래스의 평균 트랜잭션 수는 50이 된다. 부하 폭주율은 특정 트랜잭션 클래스의 부하를 균등한 부하 상태에 대한 상대 비율로 나타낸다. 예를 들어, 20%의 부하 폭주율은 폭주하지 않는 트랜잭션 클래스들의 부하가 20%(10개의 트랜잭션) 감소하고 감소된 부하의 총 합(70개의 트랜잭션)이 폭주 트랜잭션 클래스에 추가된다. 즉, 폭주 트랜잭션 클래스의 트랜잭션 수는 120이 되고 폭주하지 않는 다른 트랜잭션들의 트랜잭션 수는 40이 된다.

트랜잭션 부하가 균형적으로 분산될 경우(부하 폭주율 0%), 서로 다른 트랜잭션 클래스들에 의해서 액세스되는 모든 페이지를 한 노드의 버퍼에 캐싱할 수 없기 때문에 단일 노드의 성능이 가장 떨어진다. 낮은 버퍼 히트율은 마감기한을 위반하는 트랜잭션 수를 증가시키는 결과를 초래한다. PAR와 DACA 모두 트랜잭션 클래스를 해당 친화도 노드에 할당하여 그림 2의 (d)와 같이 높은 버퍼 히트율을 나타내었다. 부하 폭주율이 증가할 경우 PAR의 성능이 떨어진다. 그 이유는 PAR폭주 트랜잭션 클래스의 부하를 다른 노드로 분산시키지 못하기 때문에 낮은 버퍼 히트율과 초기에 고정된 컴퓨팅 자원들을 높은 부하에서도 똑같이 사용하기 때문이다. 100% 부하 폭주의 경우 PAR는 단일 노드와 동일한 성능을 나타내었다. DACA는 부하 폭주율이 증가할수록 성능이 더 좋아졌다. 그 이유는 그림 2의 (d)와 같이 부하 폭주율이 증가함에 따라 버퍼 히트율이 증가하기 때문이다. DACA는 폭주하는 트랜잭션 클래스에 노드를 추가로 할당하여 노드들 사이에 부하를 분산시킴으로써 전체 버퍼의 크기가 증가하고 따라서 버퍼 히트율도 증가한다.

4.2 실험 2 : 로크 단위에 따른 캐쉬 일관성 관리 기법의 복잡성

본 절에서는 데이타 충돌의 원인이 되는 갱신 비율(UpdatePr)을 다양하게 변화시키며 페이지와 레코드 단위의 로킹에 의한 성능을 비교한다. 그림 3의 (a)와 (b)는 부하 폭주율을 0%와 100%로 각각 설정하였을 때의 실험 결과를 나타낸다. MPL은 240으로 설정하였고, 동



(a) 마감기한 위반율 (부하 폭주율 0%) (b) 마감기한 위반율 (부하 폭주율 100%)

그림 3 실험 결과 2

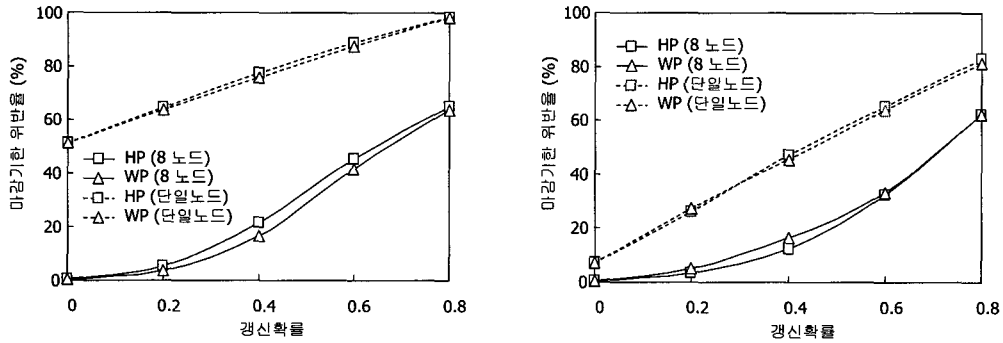
시성 제어와 트랜잭션 라우팅을 위해 각각 HP와 DACA를 수행하였다.

부하 폭주율이 0%일 경우, 로크의 단위는 성능에 크게 영향을 미치지 않는다. 이것은 트랜잭션 사이의 로크 경쟁이 희박하기 때문이다. 그러나 부하 폭주율이 100%일 경우, 모든 트랜잭션들이 동일한 트랜잭션 클래스에 속하기 때문에 동일한 데이타베이스 클러스터를 액세스하게 되고, 따라서 트랜잭션들 사이에 로크 경쟁이 빈번하게 발생한다. 이 경우 갱신 비율이 증가함에 따라 레코드 단위 로킹이 페이지 단위 로킹보다 좋은 성능을 나타내었다. 페이지 단위 로킹은 많은 트랜잭션들이 우선순위 역전에 의해 블록 되거나 철회되어 마감기한을 위반하는 트랜잭션 수가 증가되는 결과를 초래한다. 또한 레코드 단위 로킹에 의해 증가되는 캐쉬 일관성 오버헤드가 전체 성능에 그다지 큰 영향을 미치지 않는 것으로 나타났다. 본 논문을 통해 제시하지는 않지만, 갱신 비율이 낮고 동일한 페이지에서 데이타를 여러 번 액세스할 때 페이지 단위 로킹이 레코드 단위 로킹보다 좋은 성능을 보였다. 캐싱된 페이지의 일관성을 체크하기 위해 페이지 단위 로킹은 단 한번의 메시지 전송이 필요하지만, 레코드 단위 로킹은 많은 메시지 전송을 필요로 하기 때문이다.

그림 3의 (b)에서와 같이 부하 폭주율이 높을 때 SD 클러스터가 단일 노드 시스템 보다 갱신 비율의 변화에 좀더 민감하게 반응한다. 이것은 높은 부하 폭주율에서 갱신 비율에 따라 캐쉬 일관성 관리 기법의 오버헤드가 민감하게 증가하기 때문이다. 즉, 갱신 비율과 부하 폭주율이 높을 경우 노드들 사이에 버퍼 무효화 빈도가 증가하고 따라서 더 많은 페이지가 노드들 사이에서 전송되기 때문이다.

4.3 실험 3 : 동시성 제어 기법

마지막으로 실시간 동시성 제어 기법인 HP와 WP의 성능을 갱신 비율을 변화시키면서 비교한다. 이를 위해 MPL은 240으로 설정하고 트랜잭션 라우팅 기법은 DACA를 수행하였다. 로크의 단위는 로크 충돌을 좀더 빈번하게 발생시키기 위하여 페이지로 설정하였다. 그림 4의 (a)와 (b)는 부하 폭주율이 각각 0%와 100%일 때의 결과를 나타낸다.



(a) 마감기한 위반율 (부하 폭주율 0%) (b) 마감기한 위반율 (부하 폭주율 100%)

그림 4 실험 결과 3

부하 폭주율이 0%인 경우 8개의 노드로 구성된 SD 클러스터에서 WP가 HP보다 성능이 좋은 것으로 나타났으며 그 이유는 다음과 같다. 첫째, HP의 경우 불필요한 트랜잭션 철회가 발생하고, 둘째, WP의 경우 우선순위를 상속함으로써 높은 우선순위 트랜잭션이 디스크 I/O 큐에서 대기하는 시간을 감소시키기 때문이다. 부하 폭주율이 0%인 경우 상대적으로 버퍼 히트율이 낮기 때문에 디스크 I/O 큐에서의 대기 시간을 최소화하는 것이 가장 중요하다. 부하 폭주율이 100%인 경우 HP가 WP보다 좋은 성능을 나타내었다. 이 경우 버퍼 히트율이 높기 때문에 트랜잭션을 마감기한 내에 완료하기 위해서는 데이타 경쟁을 효율적으로 해결하는 것이 가장 중요하다. 일반적으로 WP는 높은 데이타 경쟁 환경에서 나쁜 성능을 나타낸다[16]. WP는 우선순위가 낮은 트랜잭션이 우선순위가 높은 트랜잭션을 반복적으로 지연시키고, 우선순위 상속은 많은 트랜잭션이 동일한 우선순위로 실행되는 결과를 초래한다. 그 결과, 높은 우선순위의 트랜잭션이 우선적으로 처리를 받을 수 없게 된다.

5. 결 론

실시간 트랜잭션 처리와 SD 클러스터는 각각 독립적으로 폭 넓은 연구가 이루어졌지만, 클러스터를 이용한 실시간 트랜잭션 처리에 관한 연구는 거의 이루어지지 않았다. 클러스터를 이용할 경우, 높은 가용성을 가지는 실시간 데이타베이스 서비스를 보장할 수 있다. 뿐만 아니라, 노드들 사이의 병렬성과 데이타의 효율적인 캐싱은 디스크 IO를 감소시켜 고성능 실시간 트랜잭션 처리가 가능하다. 본 논문에서는 SD-RTDBS의 실험 모형을 개발하여 실시간 트랜잭션 처리 기법과 SD 클러스터 기법들 간의 관계를 실험하였다.

본 논문에서 수행한 실험을 요약하면 다음과 같다. 첫째, 실시간 트랜잭션을 처리하는 SD 클러스터에서 동시에 실행되는 트랜잭션 수와 노드의 수가 많을 경우, 트랜잭션 라우팅 정책에 의해 성능이 결정되었다. 즉, 노드간 부하 분산과 버퍼 히트율을 높일 수 있는 라우팅 정책을 사용할 경우에만 SD 클러스터의 성능이 우수한 것으로 나타났다. 비효율적인 라우팅 정책을 사용했을 경우에는 다수 개의 노드를 갖는 SD 클러스터라도 단일 노드 시스템과 성능이 큰 차이가 없는 것으로 나타났다. 따라서 실시간 SD 클러스터가 갖추어야 할 실시간 트랜잭션 라우팅 정책은 트랜잭션들의 우선순위를 만족시키면서, 노드간 부하 분산과 높은 버퍼 히트율을 지원해야 함을 알 수 있다. 둘째, 레코드 단위 로킹에서 캐쉬 일관성의 복잡성과 오버헤드가 심각하지 않았고, 대부분의 시스템 환경에서 레코드 단위 로킹이 페이지 단위 로킹보다 성능이 우수한 것으로 나타났다. 따라서 레코드 단위 로킹을 기반으로 한 기존 캐쉬 일관성 기법을 트랜잭션 우선순위를 고려한 실시간 캐쉬 일관성 기법으로 확장하는 작업이 실시간 SD 클러스터에 필요함을 알 수 있다. 마지막으로 버퍼 히트율과 데이타 충돌의 정도가 낮은 경우에 WP가 HP보다 좋은 성능을 나타냈다. 높은 데이타 경쟁 환경에서는 중앙 집중 데이타베이스 시스템과 마찬가지로 HP가 좋은 성능을 나타내었다.

본 논문의 향후 연구 계획은 다음과 같다. 먼저 본 논문에서는 실시간 트랜잭션 처리를 위한 SD 클러스터의 도입과 그 성능에 대해서 분석하였지만, 동일한 접근 방법이 SN 클러스터에서도 가능할 것으로 판단된다. 일반적인 온라인 트랜잭션 처리 분야를 대상으로 한 SD 클러스터와 SN 클러스터의 성능 평가에 대한 연구는 기존에 많이 발표되었지만[10,11], 이를 실시간 환경으로 확장한 연구는 본 연구진이 파악하기에는 아직까지 보고된 바 없다. 이런 관점에서 SN 클러스터 기반의 실시간 트랜잭션 처리 알고리즘을 개발한 후, SD 클러스터와의 성능 비교를 진행하고자 한다. 다음으로 본 논문의 실험 결과를 바탕으로 본 연구진에서는 가격이 저렴한 PC들을 SD 클러스터로 다수 개 연동하여 트랜잭션을 실시간으로 처리할 수 있는 실시간 SD 클러스터를 구축하고자 한다. 이를 위하여 트랜잭션의 우선순위를 고려한 실시간 트랜잭션 라우팅 기술과 실시간 전역 캐쉬 일관성 기법을 현재 연구 중에 있다.

참 고 문 헌

[1] K-Y. Lam and T-W. Kuo (ed.), *Real-Time Database Systems: Architecture and Techniques*, Kluwer Academic Publishers, 2000.

[2] V. Kanitkar and A. Delis, "Real-Time Processing in Client-Server Databses," *IEEE Trans. Computers*, Vol.51, No.3 pp.269-288, 2002.

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

[4] *DB2 Universal Database for OS/390 and z/OS - Data Sharing: Planning and Administration*, IBM SC26-9935-01, 2001.

[5] M. Vallath, *Oracle Real Application Clusters*, Elsevier Digital Press, 2004.

[6] H. Cho and J. Park, "Maintaining Cache Coherency in a Multisystem Data Sharing Environment," *J. Syst. Architecture*, Vol.45, No.4, pp.285-303, 1998.

[7] H. Cho, "Cache Coherency and Concurrency Control in a Multisystem Data Sharing Environment," IEICE Trans. Info. and Syst., Vol.E82-D, No.6, pp.1042-1050, 1999.

[8] 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.
[9] K. Ohn and H. Cho, "Cache Conscious Dynamic Transaction Routing in a Shared Disks Cluster," *Lecture Notes in Computer Science*, Vol.3045, pp.548-557, 2004.
[10] P. Yu and A. Dan, "Performance Analysis of Affinity Clustering on Transaction Processing Coupling Architecture," *IEEE Trans. Knowledge and Data Eng.*, Vol.6, No.5, pp.764-786, 1994.
[11] P. Yu and A. Dan, "Performance Evaluation of Transaction Processing Coupling Architectures for Handling System Dynamics," *IEEE Trans. Parallel and Distributed Syst.*, Vol.5, No.2, pp.139-153, 1994.
[12] R. Abbott and H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation," *ACM Trans. Database Syst.*, Vol.17, No.3, pp.513-560, 1992.
[13] K-Y. Lam, T-W. Kuo, B, Kao, T. Lee, R. Cheng, "Evaluation of Concurrency Control Strategies for Mixed Soft Real-Time Database Systems," *Info. Syst.*, Vol.27, No.2, pp.123-149, 2002.
[14] J. Harista and S. Seshadri, "Real-Time Index Concurrency Control," *IEEE Trans. Knowledge and Data Eng.*, Vol.12, No.3, pp.429-447, 2000.
[15] O. Ulusoy, "Analysis of Concurrency Control Protocols for Real-Time Database Systems," *Info. Sci.*, Vol.111, No.1-4, pp.19-47, 1998.
[16] J. Harita, M. Carey and M. Livny, "Data Access Scheduling in Firm Real-Time Database Sytems," *J. Real-Time Syst.*, Vol.4, No.3, pp.203-241, 1994.
[17] V. Lee and K-W. Lam, "Conflict-free Transaction Scheduling using Serialization Graph for Real-Time Databases," *J. Syst. and Software*, Vol.55, No.1, pp.57-65, 2000.
[18] H. Schwetmann, User's Guide of CSIM18 Simulation Engine, Mesquite Software, Inc. 1996.

이 상 호
2003년 영남대학교 컴퓨터공학과 학사
2003년~현재 영남대학교 컴퓨터공학과 석사과정. 관심분야는 분산/병렬 데이타베이스, 실시간 트랜잭션 처리 등

온 경 오
1999년 영남대학교 컴퓨터공학과 학사
2001년 영남대학교 컴퓨터공학과 석사
2004년 영남대학교 컴퓨터공학과 박사
2004년~현재 영남대학교 정보통신 연구소 연구교수. 관심분야는 분산/병렬 데이타베이스, 트랜잭션 처리, 실시간 데이타베이스 등

조 행 래
1988년 서울대학교 컴퓨터공학과 학사
1990년 한국과학기술원 전산학과 석사
1995년 한국과학기술원 전산학과 박사
1995년~현재 영남대학교 전자정보공학부 부교수. 관심분야는 분산/병렬 데이타베이스, 트랜잭션 처리, DBMS 개발 등