COSMOS/MT: 객체 저장 시스템 COSMOS를 위한 멀티프로세스/멀티쓰레드 모델의 설계 및 구현

김이른[0] 이영구 장지웅 황규영
한국과학기술원 전자전산학과 전산학 전공/첨단정보기술연구센터
{early, yklee, duribaba, kywhang}@mozart.kaist.ac.kr

Design and Implementation of a Multi-Process/Multi-Thread Model for the COSMOS Object Storage System

Yi-Reun Kim[0] Young-Koo Lee Ji-Woong Chang Kyu-Young Whang
Department of Electrical Engineering & Computer Science
Division of Computer Science and
Advanced Information Technology Research Center
Korean Advanced Institute of Science and Technology

요 약

다수 사용자를 지원하는 프로그램에서 쓰레드의 중요성이 증가함에 따라 데이타베이스 관리 시스템의 하부구조인 객체 저장 시스템들도 쓰레드를 이용하도록 확장되고 있다. 기존의 프로세스/쓰레드 모델는 멀티프로세스/단일쓰레드 모델, 단일프로세스/멀티쓰레드 모델, 그리고 멀티프로세스/멀티쓰레드 모델로 분류할 수 있다. 이 중 멀티프로세스/멀티쓰레드 모델은 다른 모델들을 포괄할 수 있는 일반적인 형태의 구조이다.

본 논문에서는 멀티프로세스/단일쓰레드 모델로 개발된 객체 저장 시스템 COSMOS 를 멀티프로세스/멀티쓰레드 모델로 확장한 COSMOS/MT 를 설계하고 구현한다. 먼저 COSMOS 의 트랜잭션 컨텍스트를 분석하여 공유 트랜잭션 컨텍스트와 비공유 트랜잭션 컨텍스트로 분류한 후, 각 트랜잭션 컨텍스트의 유지방법을 제안한다. 그리고, 구현한 모델의 유용성을 보이기 위하여 TPC-A 벤치마크에 대해 성능 평가를 수행한다. 실험결과 1000 개의 클라이언트를 서비스하는 경우 COSMOS/MT 가 COSMOS 에 비하여 처리율이 최고 5 배까지 향상됨을 보인다. 마지막으로, 멀티프로세스/멀티쓰레드 모델의 성능을 결정하는 중요 요소인 프로세스 당 쓰레드 개수에 따른 성능 변화에 대하여 고찰하고, 실험을 통하여 프로세스당 쓰레드 개수에 따른 시스템의 성능 변화를 보인다.

1. 서 론

데이타베이스 관리 시스템은 서로 관련있는 데이타들의 집합인 데이타베이스를 관리하기 위한 프로그램이다[Sil94]. 인터넷이 널리 보급됨에 따라 데이타베이스 관리 시스템을 동시에 액세스하는 사용자 수가 증가하면서, 여러 사용자의 요구를 동시에 처리함으로써 빠른 응답 시간을 제공하고, 시스템 처리율이 높은 데이타베이스 관리 시스템이 요구된다.

객체 저장 시스템은 데이타베이스 관리 시스템에서 데이타를 저장하고 관리하는 서브 시스템으로 데이타베이스 관리 시스템의 성능에 큰 영향을 미치는 핵심 요소이다[Cho85]. 시스템 처리율이 높은 데이타베이스 관리 시스템이 필요함에 따라 많은 사용자의 요청을 빠르게 처리하는 고성능 객체 저장 시스템이 요구된다. 고성능 객체 저장 시스템을 만드는 방법중에는 한 프로세스(process)내에 여러 개의 쓰레드(thread)를 두는 멀티쓰레드 방법이 있다.

데이타베이스 관리 시스템에서 사용자의 작업을 처리하기 위하여 운영체제에서 제공하는 프로세스 또는 쓰레드를 사용하는 방법에 따라 멀티프로세스/단일쓰레드 모델, 단일프로세스/멀티쓰레드 모델, 그리고 멀티프로세스/멀티쓰레드 모델로 분류된다.

멀티프로세스/단일쓰레드 모델은 서버(server)내에 한개의 쓰레드를 가진 프로세스가 여러개 존재하는 구조이고, 단일프로세스/멀티쓰레드 모델은 서버내에 여러 개의 쓰레드를 가진 프로세스가 단 하나만 존재한다[Orf96]. 그리고, 멀티프로세스/멀티쓰레드 모델은 서버내에 여러 개의 쓰레드를 가진 프로세스가 여러개 존재하는 구조이다[Orf96]. 이 중 멀티프로세스/멀티쓰레드 모델은 다른 모델들을 포괄할 수 있는 일반적인 형태의 구조이다. 본 논문에서 서버는 데이타베이스 관리 시스템이 동작하는 컴퓨터를 가리킨다.

* 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음.

본 논문에서는 한국과학기술원 데이타베이스 및 멀티미디어 연구실에서 개발한 멀티프로세스/단일쓰레드 모델인 객체 저장시스템 COSMOS 를 멀티프로세스/멀티쓰레드 모델인 COSMOS/MT 로 설계 및 구현한다. COSMOS 의 트랜잭션 컨텍스트를 분석하여 공유 트랜잭션 컨텍스트와 비공유 트랜잭션 컨텍스트로 분류한 후, 분류한 트랜잭션 컨텍스트들에 대한 컨텍스트 유지 방법을 제안한다. 그리고, 제안한 모델의 유용성을 보이기 위하여 데이타베이스 관리 시스템의 성능을 측정하는데 널리 사용되는 TPC-A 벤치마크[Gra93a]에 대해 성능 평가를 수행한다. 실험결과 1000 개의 클라이언트를 서비스하는 경우 COSMOS/MT 가 COSMOS 에 비하여 처리율이 최고 5 배까지 향상됨을 보인다. 마지막으로, 멀티프로세스/멀티쓰레드 모델의 성능을 결정하는 중요 요소인 프로세스 당 쓰레드 개수에 따른 성능 변화에 대하여 고찰하고, 실험을 통하여 프로세스당 쓰레드 개수에 따른 시스템의 성능 변화를 보인다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 데이타베이스 관리 시스템들의 프로세스/쓰레드 모델을 소개하고, 제 3 장에서는 COSMOS/MT 의 설계 및 구현 방법을 설명한다. 제 4 장에서는 멀티프로세스/멀티쓰레드 모델로 구현된 COSMOS/MT 의 성능을 측정한다. 마지막으로, 제 5 장에서는 결론을 내리고 앞으로의 연구 방향을 제시한다.

2. 연구 배경

본 장에서는 데이타베이스 관리 시스템에서 프로세스와 쓰레드를 사용하는 방법에 따른 프로세스/쓰레드 모델을 설명한다. 제 2.1 절에서는 멀티프로세스/단일쓰레드 모델에 대해서 설명하고, 제 2.2 절에서는 단일프로세스/멀티쓰레드 모델에 대해서 설명한다. 그리고, 2.3 절에서는 멀티프로세스/멀티쓰레드 모델에 대해서 설명한다.

2.1 멀티프로세스/단일쓰레드 모델

멀티프로세스/단일쓰레드 모델은 서버내에 한개의 쓰레드를 가지는 프로세스가 여러 개 존재하여, 클라이언트(client)마다 하나의 프로세스가 할당된다[Orf96]. 즉, 클라이언트 개수만큼의 프로세스가 서버내에 존재한다. 멀티프로세스/단일쓰레드 모델을 갖는 데이타베이스 관리 시스템으로는 DB2/2 V1, Informix, 그리고 Oracle 6 이 있다[Orf96].

이 모델의 장점은 각 프로세스들이 서로 다른 메모리 주소 공간을 사용하므로 프로세스마다 독립된 데이타를 유지하기 쉽다는 것이다. 이 모델에서는 클라이언트당 하나의 프로세스가 할당되므로 각 트랜잭션들의 정보가 서로 다른 프로세스의 메모리 주소 공간에 저장되는데, 각 프로세스의 메모리 주소 공간은 운영체제에 의해서 자동적으로 보호되기 때문이다[Sil94].

그러나, 각 프로세스마다 독립된 메모리 주소 공간을 사용하기 때문에 각 프로세스들이 데이타를 서로 공유해야 하는 경우, 공유되는 데이타를 유지하는 방법이 필요하다. 그리고, 멀티쓰레드를 지원하는 모델에 비해서 운영체제 자원을 더 많이 사용한다. 왜냐하면, 클라이언트당 하나의 프로세스가 할당되므로 다른 모델에 비해 더 많은 수의 프로세스가 서버내에 생성되는데, 프로세스는 쓰레드에 비해 운영체제 자원을 더 많이 요구하기 때문이다. 마지막으로, 여러 개의 클라이언트들이 동시에 수행되는 경우, 프로세스들간의 컨텍스트 스위칭이 발생하기 때문에 오버헤드가 크다.

2.2 단일프로세스/멀티쓰레드 모델

단일프로세스/멀티쓰레드 모델은 서버내에 여러 개의 쓰레드들을 가지는 하나의 프로세스가 존재하며, 클라이언트마다 하나의 쓰레드가 할당된다[Orf96]. 즉, 클라이언트 개수만큼의 쓰레드가 서버내의 한 프로세스안에 존재한다. 단일프로세스/멀티쓰레드 모델을 갖는 데이타베이스 관리 시스템으로는 Sybase 와 MS SQL Server 가 있다[Orf96].

단일프로세스/멀티쓰레드 모델은 멀티프로세스/단일쓰레드 모델에 비해서 사용하는 운영체제 자원이 더 적다. 모든 쓰레드들이 하나의 프로세스내에 존재하므로 서버내의 모든 쓰레드들은 운영체제 자원의 일부를 서로 공유하기 때문이다. 그리고, 멀티프로세스/단일쓰레드 모델에 비해 더 높은 처리율을 갖는다. 여러 개의 클라이언트가 동시에 수행될 때, 프로세스간 컨텍스트 스위칭이 아닌 쓰레드간 컨텍스트 스위칭이 발생하므로 멀티프로세스/단일쓰레드 모델에 비해 컨텍스트 스위칭에 소비되는 시간이 더 적기 때문이다. 또한, 모든 쓰레드들이 서로 공통된 메모리 주소 공간을 사용하므로 쓰레드간에 공유되는 데이타를 유지하기 쉽다.

이 모델이 가지는 단점은 쓰레드마다 데이타를 독립적으로 유지해야 하는 경우, 독립적인 데이타를 유지하기 위한 별도의 방법이 필요하다는 것이다. 왜냐하면 모든 쓰레드들이 서로 공통된 메모리 주소 공간을 사용하므로, 모든 데이타가 서로 공유되기 때문이다. 그리고, 너무 많은 클라이언트들이 동시에 단일프로세스/멀티쓰레드 모델의 데이타베이스 관리 시스템에 접근하는 경우, 한 프로세스내에 너무 많은 쓰레드들이 생성되어 오히려 데이타베이스 관리 시스템의 성능이 저하된다.

2.3 멀티프로세스/멀티쓰레드 모델

멀티프로세스/멀티쓰레드 모델은 서버내에 여러 개의 쓰레드들을 가지는 여러 개의 프로세스가 존재하며, 클라이언트마다 하나의 쓰레드가 할당된다. 이 모델이 단일프로세스/멀티쓰레드 모델과 다른 점은 서버내에 여러개의 프로세스가 존재한다는 점이다. 멀티프로세스/멀티쓰레드 모델을 갖는 데이타베이스 관리 시스템으로는 Ingress 가 있다[Han97].

이 모델에서는 한 프로세스내에 너무 많은 쓰레드가 생성되는 경우, 새로운 프로세스를 생성시키고 새로 생성된 프로세스내에서 쓰레드를 생성하여 쓰레드들을 여러 프로세스에 분산시킬 수 있으므로 데이타베이스 관리 시스템의 성능이 떨어지지 않는다.

또한, 멀티프로세스/단일쓰레드 모델과 단일프로세스/멀티쓰레드 모델에 비해서 더 많은 트랜잭션을 동시에 처리한다. 멀티프로세스/단일쓰레드 모델에서 동시에 수행될 수 있는 트랜잭션의 개수는 서버내에 생성할 수 있는 프로세스의 개수로 제한되고, 단일프로세스/멀티쓰레드 모델에서 동시에 수행될 수 있는 트랜잭션의 개수는 한 프로세스내에 생성할 수 있는 쓰레드의 개수로 제한된다. 그러나, 멀티프로세스/멀티쓰레드 모델에서는 동시에 수행될 수 있는 트랜잭션의 개수가 "서버내에 생성할 수 있는 프로세스의 개수 x 한 프로세스내에 생성할 수 있는 쓰레드의 개수"까지 가능하기 때문에 멀티프로세스/단일쓰레드 모델과 단일프로세스/멀티쓰레드 모델에 비해서 더 많은 트랜잭션이 동시에 수행된다.

그러나, 멀티프로세스/멀티쓰레드 모델에서는 쓰레드들이 여러 프로세스내에 분산되므로 여러 프로세스내에 존재하는 쓰레드들이 데이타를 서로 공유하기 위한 별도의 방법이 필요하다. 또한, 동일한 프로세스내의 쓰레드들은 서로 동일한 메모리 주소 공간을 사용하기 때문에 쓰레드마다 독립된 데이타를 유지해야 하는 경우, 독립적인 데이타를 유지하는 방법이 필요하다.

3. COSMOS/MT 의 설계 및 구현

본 장에서는 멀티프로세스/멀티쓰레드 모델인 COSMOS/MT 의 설계 및 구현 방법을 설명한다. 제 3.1 절에서는 트랜잭션이 수행되는 동안 유지되는 트랜잭션 정보인 트랜잭션 컨텍스트에 대해 설명하고, 제 3.2 절에서는 공유 트랜잭션 컨텍스트의 유지 방법을 설명하고, 3.3 절에서는 비공유 트랜잭션 컨텍스트의 유지 방법을 설명한다.

3.1 트랜잭션 컨텍스트

트랜잭션 컨텍스트는 트랜잭션이 수행되는 동안 유지되어야 하는 정보를 뜻한다[Gra93b]. 트랜잭션 컨텍스트에는 동시성 제어를 위한 로크(lock), 데이타베이스의 데이타를 저장하고 있는 버퍼(buffer), 그리고, 화일내에서 트랜잭션이 데이타를 액세스하고 있는 위치를 나타내는 커서(cursor)등이 있다[Gra93b].

트랜잭션 컨텍스트는 트랜잭션들간의 공유 여부에 따라 공유 트랜잭션 컨텍스트와 비공유 트랜잭션 컨텍스트로 분류된다. 공유 트랜잭션 컨텍스트는 트랜잭션들이 서로 공유해야 하는 트랜잭션 컨텍스트를 의미한다. 공유 트랜잭션 컨텍스트의 예에는 로크, 버퍼가 있다.

비공유 트랜잭션 컨텍스트는 트랜잭션마다 독립적으로 유지해야 하는 트랜잭션 컨텍스트를 의미한다. 비공유 트랜잭션 컨텍스트의 예에는 커서가 존재한다. 커서는 트랜잭션이 데이타베이스의 데이타를 액세스하고 있는 위치이다.

멀티프로세스/단일쓰레드 모델인 COSMOS 를 멀티프로세스/멀티쓰레드 모델로 확장하기 위해서는 COSMOS 에서 관리되는 트랜잭션 컨텍스트들을 공유 트랜잭션 컨텍스트와 비공유 트랜잭션 컨텍스트로 구분하여야 한다. 그리고, 각각의 트랜잭션 컨텍스트에 대한 유지 방법을 제공하여야 한다.

3.2 공유 트랜잭션 컨텍스트의 유지방법

프로세스는 각각 독립적인 메모리 주소 공간을 사용하기 때문에 다른 프로세스내의 쓰레드들간에는 기본적으로 데이타가 공유되지 않는다. 그러나, 공유 트랜잭션 컨텍스트는 모든 쓰레드들이 서로 공유해야 하는 컨텍스트이므로 다른 프로세스내의 쓰레드들간에도 공유 트랜잭션 컨텍스트가 서로 공유되도록 설계되어야 한다. 또한, 여러 쓰레드들이 동시에 공유 트랜잭션 컨텍스트를 액세스 하더라도 공유 트랜잭션 컨텍스트의 데이타가 파손되지 않도록 하기 위해서는 상호 배제와 동기화 기능이 필요하다. 상호 배제란 한 순간에 하나의 쓰레드만이 공유 트랜잭션 컨텍스트를 액세스하게 하는 기능이며, 동기화란 공유 트랜잭션 컨텍스트에 대한 액세스가 순서적으로 이루어질 수 있도록 보장해 주는 기능을 말한다.

COSMOS/MT 에서는 공유 트랜잭션 컨텍스트를 유지하기 위해 공유 메모리(shared memory) 기능을 사용하고, 공유 트랜잭션 컨텍스트에 대한 상호 배제 및 동기화를 제공하기 위해서 래치(latch)를 구현하여 사용한다. 래치의 구현을 위하여 원자적 명령과 쓰레드 패키지에서 제공하는 쓰레드 세마포어(semaphore)를 이용하는 방식[Gra93b]을 사용한다. 이 방식은 상호 배제를 위하여 원자적 명령을 이용하고, 동기화를 위해서 쓰레드 세마포어를 이용하는 방식이다. COSMOS/MT 에서는 쓰레드들이 여러 프로세스에 분산되어 있기 때문에 한 프로세스내의 쓰레드들간의 동기화뿐만 아니라 서로 다른 프로세스내에 존재하는 쓰레드들간의 동기화까지 지원하는 쓰레드 세마포어를 사용한다.

3.3 비공유 트랜잭션 컨텍스트의 유지방법

한 프로세스내의 쓰레드들은 서로 동일한 메모리 주소 공간을 사용하므로 독립적으로 유지되어야 하는 비공유 트랜잭션 컨텍스트가 다른 트랜잭션에 의하여 갱신될 수 있다. 따라서, 쓰레드마다 독립적으로 비공유 트랜잭션 컨텍스트를 유지하는 방법이 제공되어야 한다.

COSMOS/MT 에서는 비공유 트랜잭션 컨텍스트를 각 프로세스내의 고유 메모리 주소 공간에 저장하고, 쓰레드는 고유 메모리 주소 공간에 존재하는 비공유 트랜잭션 컨텍스트를 핸들(handle)을 사용하여 서로 독립적으로 유지한다. 핸들이란 현재 상태를 가지고 있는 메모리 영역에 대한 인덱스이다[Ora99].

쓰레드마다 유지해야 하는 비공유 트랜잭션 컨텍스트들은 프로세스내의 공유 메모리 주소 공간에 존재하는 per-thread table 에서 관리된다. per-thread table 은 각 엔트리가 하나의 비공유 트랜잭션 컨텍스트를 갖는 엔트리들의 배열이다. COSMOS/MT 에서는 각 쓰레드들이 처리해야 하는 비공유 트랜잭션 컨텍스트를 저장하고 있는 per-thread table 의 엔트리의 인덱스를 핸들로 사용하여 자신이 처리해야 하는 비공유 트랜잭션 컨텍스트를 서로 독립적으로 액세스한다.

5. 실험 결과

본 장에서는 프로세스당 쓰레드의 개수에 따른 성능 평가 결과를 제시한다. 제 5.1 절에서 실험 모델을 설명하고, 제 5.2 절에서 실험 결과를 설명한다.

5.1 실험 모델

본 실험의 목적은 프로세스/쓰레드 모델에 따른 객체 저장 시스템의 성능 평가이다. 실험에서 사용되는 실험 모델은 TPC-A 벤치마크[Gra93a]를 바탕으로 설계된 것이다. 실험 모델은 클라이언트 부분과 서버로 구성된다. 클라이언트 부분은 총 N 개($50 \le N \le 1000$)의 클라이언트들로 구성되며, 1200 초 동안 서버(COSMOS/MT)에 트랜잭션을 요청한다.

실험에 사용되는 트랜잭션은 은행에서 발생하는 트랜잭션을 모델링한 것으로 READ-ONLY 트랜잭션과 TPC-A 트랜잭션의 두가지가 있다. READ-ONLY 트랜잭션은 READ 연산으로만 이루어져 있으며, TPC-A 트랜잭션은 READ 연산과 WRITE 연산의 비율이 1 : 4 이다. 그리고, 실험에 사용되는 데이타베이스 크기는 클라이언트의 개수에 비례하여 증가한다. 클라이언트 10 개당 전체 데이타베이스의 크기는 10M 이다.

5.2 실험 결과

본 실험의 성능 평가 척도는 트랜잭션의 처리율(throughput)과 트랜잭션의 평균 응답 시간(response time)이다. 처리율은 1 분동안 처리한 트랜잭션의 개수이며, tpm(transaction per minute)을 단위로 사용한다. 제 5.2.1 절에서는 각 프로세스/쓰레드 모델간의 성능을 비교하고, 제 5.2.2 절에서는 프로세스당 쓰레드 개수의 변화에 따른 성능을 비교한다.

5.2.1 프로세스/쓰레드 모델간의 성능 비교

실험에 사용된 프로세스/쓰레드 모델은 멀티프로세스/단일쓰레드 모델, 프로세스당 쓰레드 개수가 50 개인 멀티프로세스/멀티쓰레드 모델, 그리고 단일프로세스/멀티쓰레드 모델이다. 각 프로세스/쓰레드 모델에 대해서 READ-ONLY 트랜잭션과 TPC-A 트랜잭션의 처리율 및 평균 응답 시간을 비교한다.

실험 결과 전체적으로 멀티프로세스/멀티쓰레드 모델이 다른 두 모델에 비해서 처리율이 더 높고, 단일프로세스/멀티쓰레드 모델이 멀티프로세스/단일쓰레드 모델보다 처리율이 더 높게 나타난다. 클라이언트의 개수가 1000 개일 경우 READ-ONLY 트랜잭션에서는 멀티프로세스/멀티쓰레드 모델의 처리율이 멀티프로세스/단일쓰레드 모델의 처리율에 비해 4.30 배가 향상되고, 단일프로세스/멀티쓰레드 모델의 처리율에 비해 1.12 배가 향상된 것을 볼 수 있고, TPC-A 트랜잭션에서는 멀티프로세스/멀티쓰레드 모델의 처리율이 멀티프로세스/단일쓰레드 모델의 처리율에 비해 5.06 배가 향상되고, 단일프로세스/멀티쓰레드 모델의 처리율에 비해 1.13 배가 향상된 것을 볼 수 있다. 특히, 클라이언트의 개수가 400 개 이상일때 단일프로세스/멀티쓰레드 모델은 처리율이 급격히 저하된 반면 멀티프로세스/멀티쓰레드 모델과 단일프로세스/멀티쓰레드 모델은 800 개 이상의 클라이언트가 존재할 때에야 처리율이 조금씩 감소한다. 따라서, 멀티프로세스/멀티쓰레드 모델이 멀티프로세스/단일쓰레드 모델과 단일프로세스/멀티쓰레드 모델보다 우수하다는 것을 알 수 있다.

5.2.1 프로세스/쓰레드 모델간의 성능 비교

본 실험에서는 하나의 프로세스에 존재하는 쓰레드 개수의 최적값을 측정하기 위하여 클라이언트의 개수를 120 개로 고정하고 한 프로세스에 속하는 쓰레드의 개수를 변화시키면서 READ-ONLY 트랜잭션과 TPC-A 트랜잭션의 처리율을 측정한다.

실험 결과 READ-ONLY 트랜잭션과 TPC-A 트랜잭션의 경우 프로세스당 쓰레드 개수가 각각 30 개, 10 개가 될때까지는 성능이 증가하다가 그 이후에는 성능이 오히려 감소하는 것을 볼 수 있다. READ-ONLY 트랜잭션의 경우 프로세스당 쓰레드 개수가 30 개인 경우가 프로세스당 쓰레드 개수가 1 개인 경우와 120 개인 경우에 비해 처리율이 각각 3.04 배와 1.80 배가 향상된 것을 볼 수 있다. 또한 TPC-A 트랜잭션의 경우 프로세스당 쓰레드 개수가 10 개인 경우가 프로세스당 쓰레드 개수가 1 개와 120 개인 경우에 비해 처리율이 각각 25%와 21%가 향상된 것을 볼 수 있다. 프로세스당 쓰레드 개수가 1 개인 경우는 멀티프로세스/단일쓰레드 모델을 의미하고, 프로세스당 쓰레드 개수가 120 개인 경우는 단일프로세스/멀티쓰레드 모델을 의미한다.

초기에 프로세스당 쓰레드 개수가 증가함에 따라 처리율이 증가하는 이유는 프로세스당 쓰레드 개수가 증가함에 따라 프로세스간의 컨텍스트 스위칭보다는 쓰레드간의 컨텍스트 스위칭의 비율이 더 높아지는데, 쓰레드간의 컨텍스트 스위칭은 프로세스간의 컨텍스트 스위칭보다 더 빠르게 일어나므로 단위 시간동안 더 많은 트랜잭션을 처리하기 때문이다. 그리고, 프로세스당 쓰레드 개수가 각각 30 개, 10 개보다 많은 경우에 처리율이 감소하는 이유는 한 프로세스내에 너무 많은 쓰레드가 생성됨에 따라 쓰레드를 관리하는데 드는 오버헤드가 증가하기 때문이다.

6. 결 론

본 논문에서는 멀티프로세스/단일쓰레드 모델인 COSMOS 가 보다 많은 사용자의 요청을 보다 빨리 처리할 수 있도록하기 위하여 멀티프로세스/멀티쓰레드 모델인 COSMOS/MT 로 확장하여 설계하고 구현하였다. 본 논문의 공헌은 다음과 같다.

COSMOS 의 트랜잭션 컨텍스트를 분석하여 공유 트랜잭션 컨텍스트와 비공유 트랜잭션 컨텍스트로 분류한 후, 공유 메모리를 이용한 공유 트랜잭션 컨텍스트 유지 방법과 핸들을 사용한 비공유 트랜잭션 컨텍스트 유지 방법을 제안하였다. 그리고 실험을 통하여 성능을 평가하여 COSMOS/MT 가 COSMOS 에 비하여 처리율 및 평균 응답 시간이 크게 향상되었음을 보였다. 또한, 멀티프로세스/멀티쓰레드 모델의 성능을 결정하는 중요 요소인 프로세스당 쓰레드 개수에 따른 성능 변화를 실험하여 최적의 프로세스당 쓰레드 개수를 구하였다.

참고 문헌

[Cho85] Chou, H-T., DeWitt, D., Katz, T., and Klug, A., ``Design and Implementation of the Wisconsin Storage System,'' In *Software Practice and Experience*, Vol. 15, No. 10, pp. 942--963, 1985.

[Gra93a] Gray, J., *The Benchmark Handbook for Database and Transaction Processing Systems*, Morgan Kaufmann, 2nd Ed., 1993.

[Gra93b] Gray, J. and Reuter, A., *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1993.

[Han97] Hann, R., Ingres Frequently Asked Questions, *http://www.cs.mu.oz.au/~yuan/Ingres/ingres.faq.html*, 1997.

[Ora99] Leverenz, L. et al., *Oracle8i Concepts Release 8.1.5*, Oracle, Feb. 1999.

[Orf96] Orfali, R., Harkey, D., and Edwards, J., *The Essential Client/Sever Survival Guide*, John Wiley & Sons, 2nd Ed., 1996.

[Sil94] Silberschatz, A. and Peter, B. G., *Operating System Concepts*, Addison-Wesley, 4th Ed., 1994.