TPC-W 성능 평가에서의 데이타베이스 시스템 성능 인자 튜닝

(Database System Parameter Tuning in the TPC-W Benchmark)

류 문 수 † 정 회 진 †† 이 상 호 †††
(Moon Soo Ryu) (Hoe Jin Jeong) (Sang Ho Lee)

요 약 대량의 데이타를 관리하는 현대 데이타베이스 시스템 환경에서는 데이타베이스 시스템 튜닝에 대한 중요성이 증가하고 있다. 특히, 데이타베이스 시스템 성능 인자(performance parameters)를 시스템 부하에 따라 적절하게 튜닝하여야 한다. 본 논문에서는 TPC-W 환경에서 데이타베이스 시스템의 단일 성능 인자 튜닝 전략 2가지(처리량 평가 방법, 응답시간 평가 방법)를 제시한다. 효과성을 입증하기 위하여 제시한 튜닝 전략은 두 개의 상용 데이타베이스 시스템에 적용하였다. 실험 결과는 제시된 튜닝 전략이 성능 향상에 기여함을 보인다.

키워드 : 데이타베이스 시스템 튜닝, 성능 인자 튜닝, TPC-W

Abstract There have been an emerging interests in the importance of database tuning techniques under modern database environments in which very large-scale data should be managed. In particular, database performance parameters should be tuned to reflect system loads appropriately. This paper presents two parameter tuning strategies, namely throughput-based and response-time-based, which tune each performance parameter accordingly. The proposed techniques are applied to two commercial database systems in the TPC-W benchmark to see the effectiveness of those methods. The results show that they can help improve system performance considerably.

Key words : database system tuning, parameter tuning, TPC-W

1. 서 론

정보 기술의 중추적인 역할을 담당하고 있는 데이타베이스 시스템은 데이타 웨어하우스(data warehouse)와 전자 상거래(electric commerce) 같은 다양한 분야에서 보편적으로 사용되고 있다. 데이타베이스 시스템이 관리하는 데이타 양은 대폭적으로 증가하고 있으며, 또한 데이타베이스 시스템이 지원하는 기능은 다양화되고 있다. 이로 인해 데이타베이스 시스템 튜닝의 중요성이 부각되고 있다[1]. 데이타베이스 튜닝은 데이타베이스 응용 프로그램이 더욱 빠르게 수행되도록 하는 관련 작업을 의미하며, '더욱 빠르게 수행된다'는 것은 데이타베이스 응용 프로그램의 처리량이 많아지는 것을 의미한다[2].

데이타베이스 시스템에서의 튜닝 대상 요소들로 질의, 인덱스, 성능 인자(performance parameters) 뿐만 아니라 하드웨어의 구성과 운영체제의 특성까지 폭넓게 고려할 수 있다[3]. 데이타베이스 튜닝 단계는 데이타 설계, 응용 프로그램, 메모리, I/O, 경합(contention), 운영체제 등으로 구분 할 수 있다[4].

데이타 설계 튜닝은 테이블과 칼럼 속성을 정의하고, 테이블 간의 관계를 설정하고, 주 키와 외래 키를 생성하고, 다양한 인덱스 생성과 클러스터 사용 여부 등을 결정하는 단계이다. 이 단계의 튜닝은 데이타가 중복되지 않도록 테이블의 정규화 및 비정규화를 결정해야 하고, 데이타에 대한 접근 경합이 발생하지 않도록 설계하는 것이다. 응용 프로그램 튜닝은 응용 프로그램(예를 들면, 의사 결정 지원 시스템, 온라인 트랜잭션 처리 등)의 특성에 따라 해당 데이타베이스 시스템의 최적기(optimizer) 모드를 선택하고, 성능 저하를 유발시키는

· 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌습니다.
† 정 회 원 : 핸디소프트 공동개발팀 연구원
moonsuda@dreamwiz.com
†† 학생회원 : 숭실대학교 컴퓨터학과
sinclear@dreamwiz.com
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
shlee@comp.ssu.ac.kr
논문접수 : 2003년 7월 21일
심사완료 : 2004년 4월 27일

비효율적인 질의를 재작성하여 성능 향상을 추구하는 단계이다. 메모리 튜닝은 데이타베이스 시스템이 관리하는 공유 메모리, 버퍼 캐시, 정렬, 해시 등의 메모리 영역을 적절히 할당하여 성능 향상을 도모하는 단계이다. 이 단계 튜닝에서는 메모리 적중률이 해당 데이타베이스 시스템에서 제시하는 적중률 범위 내에 포함될 수 있도록 크기를 할당해야 한다. I/O 튜닝은 테이블 저장 공간, 로그 파일의 구성, 체크 포인트 발생 빈도 등에 대해서 결정하는 단계이다. 이 단계 튜닝은 디스크 별로 데이타를 분산하여 I/O 발생량을 감소시켜 디스크 경합이 최소화 되도록 하고, 최적의 데이타 접근을 위한 데이타 블록(block)의 구조를 설정하는 것이다. 메모리와 I/O 튜닝은 해당 데이타베이스 시스템의 성능 인자를 정적 또는 동적으로 튜닝하는 방법으로 이루어진다. 경합 튜닝은 다중 사용자 환경에서 동시성 제어를 위한 잠금(lock)과 래치(latch)가 요구될 때 프로세스 간의 경합을 최소화하는 단계이다. 이 단계에서는 잠금에 대한 유형과 모드를 결정하고 교착 상태를 해소하는 것이며, 래치에 대한 튜닝은 데이타 버퍼 캐시에 대한 충분한 래치가 있는지 확인하여 해당 데이타베이스 시스템에서 제시하는 값에 포함될 수 있어야 한다. 운영체제 튜닝은 데이타베이스 시스템이 운영되는 특정 운영체제의 특성을 고려한 튜닝 단계이다. 이 단계의 튜닝은 운영체제에 따른 버퍼 캐시의 크기, 프로세스가 관리하는 메모리 크기 등을 결정하는 것이다.

데이타베이스 튜닝 및 관련 시스템의 성능 향상을 위한 연구는 산업계 및 학계를 중심으로 진행되고 있다. [3]은 관계형 데이타베이스 시스템과 객체 관계형 데이타베이스 시스템에 적용될 수 있는 튜닝 도구를 제안하였다. 특히, 인덱스 생성과 선택 추출 질의의 선택율(selectivity) 변화에 따른 질의 평균 응답 시간을 분석하여 인덱스 생성에 대한 지침을 제시한다. 이 도구는 10개의 카테고리에 65개의 질의어와 확장성(scalability)을 제공하며 각 질의에 대한 질의 수행 시간 결과를 보이고 있다.

[5]는 대용량 데이타베이스 환경에서 데이타베이스 관리자가 올바른 인덱스를 선택할 수 있도록 인덱스 분석 도구를 개발하였다. 이 도구는 TPC-D 성능 평가에서의 낮은 처리 비용 및 빠른 응답 시간에 대한 사용자 요구를 만족시키기 위해 개발되었다. 이 도구는 사용자의 질의 유형에 따라 실제 또는 가상 인덱스를 양적으로 분석하여 데이타베이스 관리자에게 가장 낮은 비용의 인덱스를 추천한다. 실험 결과 단일 칼럼에 대한 인덱스를 추가하면 약 5% 정도 비용이 감소하고, 다중 칼럼에 대한 인덱스를 추가하면 약 18% 정도 비용이 감소한다.

[6]은 사용자의 QoS(quality of service) 보장을 위한 전자 상거래 시스템의 모니터링 방법 및 웹 서버와 응용 프로그램 서버에서의 성능 인자 튜닝 방법을 제시하고 있다. 실험 대상 성능 인자는 웹 서버와 응용 프로그램 서버의 쓰레드 개수, 쓰레드 당 처리할 수 있는 큐의 크기이다. 측정 요소는 처리량, 응답 시간, 요청 거부 확률이다. 실험 수행 시간 마다 각 측정 요소를 측정하여 [6]에서 제시하는 QoS 값을 구한 후 이 값이 이전 QoS 값보다 감소되면 성능 인자의 크기를 힐 클라이밍(hill climbing) 알고리즘을 통해서 자동 변경하는 자체 튜닝(self-tuning) 방법을 제시하고 있다. 그리고 단일 계층 모델(single tier model)을 확장하여 다중 계층 모델(multiple tier model)에서의 성능 측정 접근법을 제시한다. 실험 결과, 자체 튜닝을 적용하지 않은 경우 웹 사용자의 수의 증감에 따라 QoS 값이 급격히 감소하지만 자체 튜닝을 적용하는 경우 QoS 값이 완만히 증감한다.

TPC(Transaction Processing Performance Council)[7]의 TPC-W는 대표적인 컴퓨팅 패러다임인 웹 환경을 대상으로 하는 성능 평가 도구로 산업계의 표준으로 널리 사용되고 있다. 본 논문은 TPC-W 성능 평가[8]에서의 데이타베이스 시스템 성능 향상을 위한 성능 인자의 튜닝 전략을 제시하고, 실험을 통해 튜닝 전략의 정당성을 평가한다. 구체적으로, 본 논문은 2가지 튜닝 전략을 제시하고, 각 튜닝 전략을 두 개의 상용 데이타베이스 시스템 X와 Y(라이선스(license) 문제로 인해 구체적인 제품명을 명시할 수 없음)에서 각각 세 개의 성능 인자에 적용하여 성능 향상 여부를 확인하고 분석하여 그 효과성을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 TPC-W의 전체적인 설명과 실험을 위한 성능 인자의 두 가지 튜닝 전략을 소개하고, 3장에서는 시스템 X와 시스템 Y에서의 튜닝 전략에 따른 실험 결과 기술 및 분석을 수행하고, 4장에서는 결론과 향후 계획에 대해 언급한다.

2. TPC-W와 성능 인자 튜닝 전략

2.1 TPC-W

TPC-W는 웹 환경에서의 데이타베이스 시스템에 대한 성능 평가의 산업 표준이며 성능 평가 방법론에 대한 매우 정확한 사양을 명시하고 있다. TPC-W는 인터넷 전자 서점의 전자 상거래 시스템을 시뮬레이션하고 있으며, 웹 상호작용(web interaction)을 기반으로 동작한다.

웹 상호작용은 사용자가 웹 서버에 대한 최초 요청에서부터 최종 응답까지에 대한 일련의 작업이고, 총 14개의 웹 상호작용을 TPC-W에서는 정의하고 있다. TPC-W의 성능 측정 요소는 측정 시간동안 처리되는

웹 상호작용의 개수를 표시하는 WIPS(web interaction per second)이며, 다음의 제약 조건을 만족하여야 유효한 측정값으로 인정된다. 이 제약 조건은 사용자 브라우저가 14개인 경우에는 초당 1~2개, 28개인 경우에는 초당 2~4개의 웹 상호작용이 수행되어야만 만족됨을 의미한다[8].

(EB개수/14) < WIPS < (EB개수/7)

EB(emulated browser)는 사용자 브라우저를 의미하며, TPC-W에서의 작업부하로 사용된다. EB는 사용자 세션 최소 유지 시간(user session minimum duration, USMD)동안 수행되고, 웹 서버로부터 요청한 응답을 받으면 다음 웹 요청을 하기 전까지 사고 시간(think time)을 가짐으로 실세계 웹 환경을 시뮬레이션하고 있다.

TPC-W는 8개의 기본 테이블("ADDRESS", "AUTHOR", "CC_XACTS", "COUNTRY", "CUSTOMER", "ITEM", "ORDERS", "ORDER_LINE")로 구성되어 있다. 그림 1은 EB 개수에 따른 TPC-W 구성 테이블들의 데이타베이스 확장성(scalability)을 보여준다. "COUNTRY" 테이블의 튜플 개수는 92개로 크기가 고정되어 있고, 나머지 테이블의 크기는 EB 개수와 "ITEM" 테이블의 튜플 개수에 의해서 결정된다. "ITEM" 테이블은 그림 1에서와 같이 1,000개~10,000,000개 중 한 경우에 해당되는 튜플 수를 가질 수 있다. 예를 들면, 10개의 EB와 10,000개의 "ITEM" 테이블 튜플 개수로 실험을 수행하는 경우, "CUSTOMER" 테이블은 28,800개, "AUTHOR" 테이블은 2,500개의 튜플을 가지게 된다.

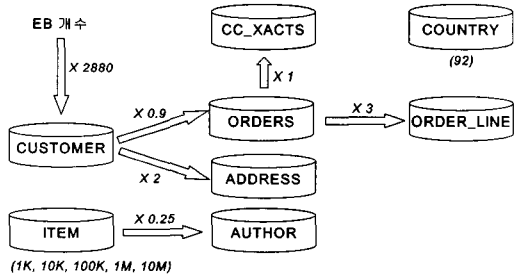

그림 1 TPC-W의 데이타베이스 확장성

웹 상호작용은 표 1의 제한된 응답시간 및 혼합비율을 만족하여야 한다. 제한된 응답시간은 수행 시간동안 처리된 전체 웹 상호작용 중 하위 90%의 웹 상호작용에 대한 평균 응답시간이고, 혼합비율은 수행 시간동안 발생되는 전체 웹 상호작용 중에서 각 웹 상호작용들이 차지하는 비율이다.

TPC-W의 전체 수행 시간은 램프 업(ramp up) 시간, 측정 시간, 반복 측정 시간으로 나뉜다. 안정된 작업부하를 준비하는 시간인 램프 업 시간은 10분 이상이며, 측정 시간 30분, 반복 측정 시간 30분을 필요로 한다. 측정 시간에 측정된 WIPS 값과 반복 측정 시간에 측정된 WIPS 값의 오차가 5% 이내여야 유효한 결과 값으로 인정된다.

2.2 성능 인자 튜닝 전략

상용 데이타베이스 관리 시스템(database management system, DBMS)에서 제공하고 있는 성능 인자의 튜닝 전략은 각 제조사들이 제시하고 있는 성능 목표치에 미달하는 경우, 관리자가 성능 인자의 크기를 임의의 크기로 적절히 변경하는 것이다. 이러한 방법은 사용자의 직감과 경험에 의존하는 임기응변(ad-hoc) 방식이다. 본 논문에서는 상용 DBMS의 튜닝 전략과는 달리 사용자의 처리량과 응답시간을 튜닝의 기준으로 정했으며, 성능 인자의 변경 크기 또한 적절한 임의의 값이 아닌 체계적으로 계산 및 변경된 값을 사용한다.

각 데이타베이스 시스템은 다양한 성능 인자를 가지며, 각 성능 인자에 대하여 기본 설정 크기를 제공한다. 기본 설정 크기는 일반적인 데이타베이스 시스템 사용 환경에서 최선의 성능으로 작업을 수행할 수 있도록 설정되어 있는 것이 보편적이다. 본 논문에서 제시하는 튜닝 전략은 이러한 기본 설정 크기를 바탕으로 하여 성능 인자 튜닝을 수행한다.

데이타베이스 부하에 따른 성능 인자 크기 설정에 관해 다음 두 가지 경우를 생각해 볼 수 있다. 첫째, 부하에 비해 성능 인자가 적정 크기보다 작은 경우, 부하를

표 1 웹 상호작용의 응답시간 제한과 혼합비율

웹 상호작용	응답시간	혼합비율	웹 상호작용	응답시간	혼합비율
Admin Confirm	20 초	0.09 %	New Products	5 초	5.00 %
Admin Request	3 초	0.10 %	Order Display	3 초	0.66 %
Best Sellers	5 초	5.00 %	Order Inquiry	3 초	0.75 %
Buy Confirm	5 초	1.20 %	Product Detail	3 초	17.00 %
Buy Request	3 초	2.60 %	Search Request	3 초	20.00 %
Customer Registration	3 초	3.00 %	Search Results	10 초	17.00 %
Home	3 초	16.00 %	Shopping Cart	3 초	11.60 %

처리하는데 많은 시간이 소요되고 시스템의 처리량도 감소한다. 둘째, 부하에 비해 성능 인자 크기가 적정 크기보다 큰 경우, 해당 성능 인자가 시스템 리소스를 과도하게 사용하여 다른 성능 인자들이 사용할 리소스가 부족하게 되어 응답시간이 늦어지고 처리량도 감소 할 수 있다[9].

두 번째 경우는 그림 2를 통해 설명된다. 그림 2는 시스템 Y에 대한 위스콘신 성능 평가[10] 결과 중에서 "NUM_DATA_BUFFERS" 성능 인자 크기 변경에 따른 선택 추출 질의(질의 번호 1, 2, 4)의 응답시간을 보인다. 실험 결과, 성능 인자가 기본 설정 크기로부터 적정 크기를 넘어서 4배까지 증가하게 되면 질의 1의 응답시간은 기본 설정 크기에서의 응답시간보다 93% 증가하였고, 질의 2의 응답시간은 37% 증가하였다. 질의 4의 응답 시간은 기본 설정 크기에 비해 거의 변화가 없었다.

```
질의 1번(no index) - 1% selection
INSERT INTO TMP
SELECT * FROM TENKTUP1
WHERE unique2 BETWEEN 0 AND 99

질의 2번(no index) - 10% selection
INSERT INTO TMP
SELECT * FROM TENKTUP1
WHERE unique2 BETWEEN 792 AND 1791

질의 4번(clustered index) - 10% selection
INSERT INTO TMP
SELECT * FROM TENKTUP1
WHERE unique2 BETWEEN 792 AND 1791
```

그림 2의 실험 결과를 통해 성능 인자 크기를 크게 설정하는 것이 반드시 성능 향상에 기여하는 것이 아니며 성능 인자의 크기 증가 시 적절한 크기로 증가되어 설정되어야 함을 알 수 있다. 부하의 변화에 따라 성능 인자의 작은 변경 크기가 부하에 대한 적절한 대응일 수도 있고, 큰 변경 크기가 적절한 대응이 될 수도 있기 때문이다. 따라서 데이타베이스 부하에 따른 적절한 성능 인자 크기를 대응시키기 위해 성능 인자 크기 변경의 범위를 결정하여야 한다. 본 논문에서는 작은 변경 크기는 기본 크기의 50%, 큰 변경 크기는 기본 크기의 200%로 정하였으며, 성능 인자의 변경 후 크기는 식 (1)에 의해서 산출된다.

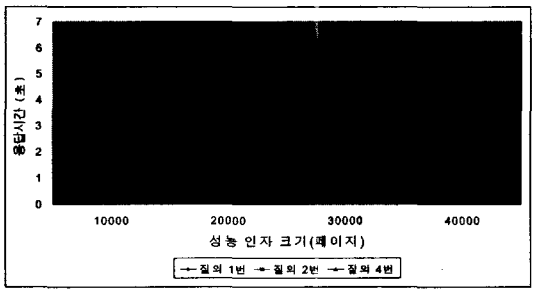

그림 2 위스콘신 성능 평가에서 시스템 Y의 응답시간 변화

$$P_{next} = P_{init} + P_{acc} + P_{inc}$$
$$P_{next} < P_{Max} \quad (1)$$

식 (1)의 P_{init}는 성능 인자의 기본 크기, P_{next}는 성능 인자의 변경 후 크기, P_{acc}는 튜닝 시점까지 증가된 성능 인자의 크기, P_{inc}는 성능 인자의 변경 크기, P_{Max}는 성능 인자의 최대 크기를 의미한다. 성능 인자의 변경 후 크기는 성능 인자의 최대 크기를 초과할 수 없다.

TPC-W를 위하여 본 논문에서 제시하는 성능 인자 변경 전략은 두 가지이다. 두 가지 변경 전략은 TPC-W 수행 중 일정 시간동안 전체 웹 상호작용의 처리량 감소가 발생하면 이를 해소하기 위해서 성능 인자 크기를 변경하는 처리량 평가 방법의 튜닝 전략과 각 웹 상호작용의 평균 응답시간이 제한 시간을 초과하면 성능 인자 크기를 변경하여 제한 시간 이내로 응답할 수 있도록 하는 응답시간 평가 방법의 튜닝 전략이다.

처리량 평가 방법에 의한 성능 인자의 튜닝 전략은, 첫째, 이전 측정 시간동안 처리된 웹 상호작용 개수보다 현재 측정 시간동안 처리된 웹 상호작용 개수가 5% 이상 적은 경우, 둘째, 이전 3회 측정 시간동안 처리된 웹 상호작용의 평균 개수보다 현재 측정 시간동안 처리된 웹 상호작용 개수가 적은 경우, 셋째, 현재 측정 시간동안 처리된 웹 상호작용 개수가 TPC-W에 명시된 EB의 개수에 따른 WIPS 값 제약 조건을 만족할 수 없는 경우 성능 인자 크기를 일정 크기만큼 변경하는 방법이다. 언급된 세 가지 성능 저하 현상 중에서 하나라도 발생하면 이 튜닝 전략이 적용된다.

첫 번째 튜닝 조건의 처리량 감소 비율은 TPC-W 사양에 따라 EB들이 상호 독립적으로 수행되어 처리량이 측정 시간마다 약 5% 까지 오차를 가질 수 있기 때문에 감소 비율을 5% 이상으로 설정하였다. 두 번째 튜닝 조건은 오차 범위 내에서 성능 저하가 지속적으로 발생하는 경우 첫 번째 튜닝 조건만으로는 성능 저하에 대한 대처를 할 수 없어 이전 3회 측정 시간동안의 평균 처리량 보다 적은 처리량을 수행한 경우 튜닝을 수행하기 위함이다. 세 번째 튜닝 조건은 본 논문에서 실험 수행 환경이 TPC-W를 기반으로 하고 있기 때문에 TPC-W 사양의 제약 사항을 만족시키기 위함이다. 예

를 들어, 20개의 EB로 5분의 측정 시간동안 수행하는 경우 최소 WIPS 값인 1.43을 만족하기 위해 측정 시간 동안 평균 428개 이상의 웹 상호작용이 처리되어야 하며, 이보다 적은 웹 상호작용을 처리하는 경우에는 성능 인자의 크기를 변경한다.

응답시간 평가 방법에 의한 성능 인자의 튜닝 전략은 "Admin Confirm" 상호 작용과 "Admin Request" 상호 작용을 제외한 웹 상호작용이 표 1의 평균 응답시간을 초과하는 경우 성능 인자 크기를 일정 크기만큼 증가하는 방법이다. "Admin Confirm"과 "Admin Request" 웹 상호 작용은 혼합 비율이 낮아 제외하였다. 한 개 이상의 웹 상호작용이 응답시간을 초과하는 경우 이 튜닝 전략이 적용된다.

3. 실험 및 결과 분석

본 연구에서는 원칙적으로 TPC-W 사양(specification)에 의거하여 TPC-W를 구현하였으나, 본 연구에 직접적인 영향을 미치지 않은 범위 내에서 다음 세 가지 사항을 미구현 또는 변경하였다. 첫째, TPC-W의 보안(security), 암호(cryptogram)와 관련된 PGE(payment gateway emulator)는 데이타베이스 시스템의 성능에 직접적인 영향을 미치지 않으므로 구현에서 제외하였다. 둘째, TPC-W 사양에서는 평균 사고 시간을 7초~8초의 시간으로 정의하고 있지만, 정의된 시간이 평균 사고 시간이므로 실질적으로는 0초~60초 사이의 무작위 사고 시간이 발생할 수 있다. 이 경우 웹 상호작용 처리 비율에 대한 편차가 클 수 있기 때문에 본 실험에서는 7초의 고정된 시간으로 사고 시간을 구현하였다. 셋째, TPC-W 사양에서는 평균 USMD(사용자 세션 최소 유지 시간)가 855초~945초로 정의하고 있으며, 평균 사고 시간과 비슷하게 실질적으로는 10초에서 7,200초(2시간) 이상까지 무작위로 USMD가 발생할 수 있다. USMD가 짧은 EB들은 TPC-W의 직접적인 작업부하가 아닌 세션 아웃(session out)과 세션 인(session in)을 빈번하게 수행하여 실험 데이타의 신뢰성을 낮출 수 있기 때문에, 본 실험에서는 USMD를 900초(15분)로 고정하여 구현하였다.

TPC-W 사양을 기반으로 한 다양한 TPC-W 구현 방법 중 본 연구에서는 성능 향상을 위하여 8개의 기본 테이블 외에 사용자 쇼핑카트(shopping cart)의 정보 관리를 위한 "SHOPPING_CART" 테이블과 쇼핑카트 내의 상품 정보 관리를 위한 "SHOPPING_CART_LIST" 테이블을 추가하였다. 쇼핑카트를 EB 내의 메모리 구조체로 구성할 경우, EB 수의 증가에 따라 RBE(remote browser emulator)의 메모리 부족 문제가 발생할 수 있고, 이로 인해 대상 시스템의 성능 저하나 TPC-W 성능 평가 시험이 불가능할 수 있기 때문이다. 실험에서는 "ITEM" 테이블이 10,000개의 튜플을 가지도록 설정하였다.

그림 3은 실험 시스템의 구성을 보인다. "웹 서버/응용 프로그램 서버"는 EB에서 요청한 웹 상호작용을 처리하여 응답하는 역할을 수행한다. 응용 프로그램 서버로는 WebLogic[11]을 사용하였고, 웹 서버로는 WebLogic에서 제공되는 웹 서버를 활용하였다. 웹 상호작용의 개발을 위해 JSP(Java Server Pages)와 자바 서블릿(Java Servlet)을 사용하였고, "웹 서버/응용 프로그램 서버"가 설치된 시스템에서 웹 상호작용이 구동된다. "RBE"는 Java 1.3을 이용하여 개발되었고, EB를 생성하고 관리하는 역할을 수행한다. "이미지 서버"는 TPC-W에서 사용되는 상품 이미지를 저장하고 있고, EB의 요청 시 이를 EB에 전달하며, 아파치 웹 서버를 이용하여 작업을 수행한다. "웹 서버/응용 프로그램 서버" 장비는 펜티엄 III 프로세서를 장착하고 있으며 운영체제는 Red Hat Linux 7.3이다. "이미지 서버" 장비는 UltraSparc2이고 운영체제는 SunOS 5.7이다.

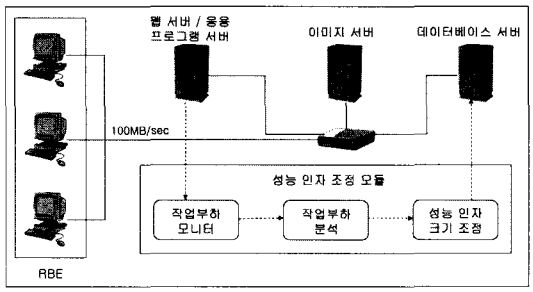

그림 3 실험 시스템 구성

"성능 인자 조정 모듈"은 실험을 수행하는 핵심 모듈로서 Java 1.3으로 개발되었으며 세 개의 세부 모듈로 나뉘어져 있다. "작업부하 모니터"는 웹 서버와 응용 프로그램 서버에서 실험 수행 시간동안 수행되는 웹 상호작용의 정보를 측정 시간마다 수집한다. "작업부하 분석"은 "작업부하 모니터"에서 수집된 정보와 데이타베이스 시스템 성능 인자의 변경 여부를 판단한다. "성능 인자 크기 조정"은 성능 인자 변경을 해야 하는 경우 성능 인자 크기를 일정 크기만큼 증가시키는 역할을 한다.

본 실험은 시스템 X와 시스템 Y로 지칭되는 상용 데이타베이스 시스템 두 개에 대하여, 각 시스템 당 세 개의 성능 인자를 대상으로 수행하였다. 성능 인자의 선정은 데이타베이스 시스템이 제공하는 다양한 성능 인자 중에서 성능 향상에 밀접한 관계가 있는 성능 인자를 중심으로 선정했다. 밀접한 관계가 있는지 여부는 여러

성능 인자들이 성능 향상에 기여하는 정도를 측정하는 기초적인 실험을 수행한 결과를 바탕으로 결정하였다.

3.1 시스템 X의 결과 및 분석

시스템 X는 상용 객체-관계형 데이타베이스 시스템이며, 단일 펜티엄 프로세서, Linux 7.3에서 운용하였다. 시스템 X에 대한 실험은 다음과 같이 진행하였다. 첫째, 성능 인자의 변경 판단은 5분마다 수행하였다. 그 이유는 측정 시간동안 응답시간 평가 방법의 튜닝 전략을 만족하기 위해 전체 웹 상호작용은 150회 이상 처리되어야 하며, 최소 EB 개수(5개)가 150회 이상을 처리하기 위해서는 약 4분 이상의 시간이 필요하기 때문이다. 둘째, EB는 15개로부터 시작하여 사용자 세션 최소 유지 시간인 15분마다 새로운 EB가 15개씩 추가된다. 셋째, 전체 실험 시간은 225분으로 하였다. 즉, 시스템 X에 대한 실험은 총 225분 동안 수행되며, 15분마다 EB가 15개씩 증가하여 210분 경과 후에는 EB 개수가 225개가 된다. 성능 인자 변경 판단은 5분마다 수행하여, 총 44번의 성능 인자 변경 판단을 수행하게 된다.

시스템 X에서 실험 대상으로 선정된 성능 인자는 "DB_CACHE_SIZE", "PGA_AGGREGATE_TARGET", "SHARED_POOL_SIZE"이다. "DB_CACHE_SIZE" 성능 인자는 버퍼 풀(pool)의 크기를 설정하고, "PGA_AGGREGATE_TARGET" 성능 인자는 정렬, 해시 조인, "group by" 등 메모리를 집중적으로 사용하는 질의에 대해 메모리 크기를 명시한다. "SHARED_POOL_SIZE" 성능 인자는 공유 커서, 내장 프로시저, SQL 문장의 파싱 등에 사용되는 공유 풀의 크기를 명시한다.

시스템 X에서 각 성능 인자의 변경 크기를 결정하는 실험은 처리량 평가 방법에 의한 튜닝 전략으로 실험을 하였으며, 실험 결과는 표 2에서 보이고 있다. "DB_CACHE_SIZE"의 기본 크기(P_{init})는 32MB, 최대 크기(P_{Max})는 1.5GB이고, "PGA_AGGREGATE_TARGET"의 기본 크기(P_{init})는 24MB, 최대 크기(P_{Max})는 1.5GB이다. "SHARED_POOL_SIZE"의 기본 크기(P_{init})는 32MB이고 최대 크기(P_{Max})는 1.5GB이다. 처리 비율은 시스템 기본 크기로 인자를 변경 하였을 때의 웹 상호작용 처리량에 대해 지정된 크기로 인자를 변경하였을 때의 웹 상호작용 처리량의 비율을 표시한 값이다. 처리 비율이 가장 높은 변경 크기가 최종 변경 크기(P_{inc})로 선정되며, "DB_CACHE_SIZE"는 32MB, "PGA_AGGREGATE_TARGET"는 24MB, "SHARED_POOL_SIZE"은 32MB이다. 이 실험에서 인자 변경 크기가 200% 경우에는 실험 수행 중 P_{Max} 크기를 초과하게 되어 실험 종료 시까지 성능 인자 크기를 변경할 수 없었기 때문에 변경 크기로 선정하지 않았다.

표 3은 시스템 X의 성능 인자 크기 변경에 대한 실험 결과를 보인다. "NO_CHANGE"는 기본 크기를 사용하여 인자 크기의 변경 없이 수행한 결과이며, 성능 비교의 기준이 된다. 시스템 X는 기본 성능 값에 비해 단일 성능 인자의 변경을 수행함으로써 대략 6~15 % 정도 성능이 향상되었다.

그림 4는 처리량 평가 방법을 적용한 실험 결과로서, "NO_CHANGE"의 결과를 포함하여 측정 시간동안 가장 많은 웹 상호작용을 처리한 최고 성능 값을 기준으로 한 처리 비율의 변화를 보인다. 측정 시간 60분까지는 성능 인자 변경을 통한 튜닝이 수행하지 않았으며 처리량이 "NO_CHANGE"의 그래프와 오차 범위 내에서 유지되었다. 측정 시간 65분~120분까지는 성능 인자들에 대한 변경이 수행되어 "NO_CHANGE"의 처리량보다 점차 많은 처리량을 보였으며, 125분 이후에서는 성능 인자의 변경을 통한 튜닝이 빈번해지고 처리량의 차이는 더욱 심화하였다.

표 2 시스템 X의 성능 인자의 변경 크기

성능 인자	50 %		100 %		200 %	
	변경 크기	처리 비율	변경 크기	처리 비율	변경 크기	처리 비율
DB_CACHE_SIZE	16 MB	0.73	32 MB	1	64 MB	-
PGA_AGGREGATE_TARGET	12 MB	0.77	24 MB	1	48 MB	-
SHARED_POOL_SIZE	16 MB	0.80	32 MB	1	64 MB	-

표 3 시스템 X의 실험 결과

평가 방법	성능 인자	처리 비율	%
NO_CHANGE	각 성능 인자 모두 포함	1	0
처리량 평가 방법	DB_CACHE_SIZE	1.1427	+ 14.27
	PGA_AGGREGATE_TARGET	1.0736	+ 7.36
	SHARED_POOL_SIZE	1.0621	+ 6.21
응답시간 평가 방법	DB_CACHE_SIZE	1.1537	+ 15.37
	PGA_AGGREGATE_TARGET	1.0612	+ 6.12
	SHARED_POOL_SIZE	1.0853	+ 8.53

실험 결과를 분석하면, "DB_CACHE_SIZE" 인자는 측정 시간 60분, 70분, 90분, 105분, 115분 등에서, "PGA_AGGREGATE" 인자는 측정 시간 60분, 70분, 90분, 95분, 105분 등에서, "SHARED_POOL_SIZE" 인자는 측정 시간 70분, 75분, 90분, 95분, 105분 등에서 성능 인자 튜닝 전략이 적용되었다. 성능 인자 튜닝 전략의 첫 번째 조건에 의하여 인자가 변경된 횟수는 측정 시간 210분까지 "DB_CACHE_SIZE" 인자는 11번, "PGA_AGGREGATE_TARGET" 인자는 13번, "SHARED_POOL_SIZE" 인자는 13번이다. 두 번째 조건에 의하여 인자가 변경된 횟수는 측정 시간 155분까지 "DB_CACHE_SIZE" 인자는 5번, "PGA_AGGREGATE_TARGET" 인자는 3번, "SHARED_POOL_SIZE" 인자는 3번이다. 세 번째 조건에 의하여 인자가 변경된 횟수는 측정 시간 205분까지 "DB_CACHE_SIZE" 인자는 4번, "PGA_AGGREGATE_TARGET" 인자는 6번, "SHARED_POOL_SIZE" 인자는 6번이다.

"DB_CACHE_SIZE" 인자는 측정 시간 115분과 135분에 처리량 감소를 해소하기 위해 인자 크기를 각각 32MB씩 증가하였고, 이후 각각 120분과 140분에서 처리량이 증가되었다. "PGA_AGGREGATE_TARGET" 인자는 측정 시간 150분과 165분에서 처리량이 감소되어 인자 크기를 24MB 증가하였고, 이후 155분과 170분에서 처리량이 증가되었다. "SHARED_POOL_SIZE" 인자는 측정 시간 175분과 200분에서 처리량 감소되어 인자 크기를 32MB 증가하였고, 이후 180분과 205분에서 처리량이 증가되었다.

그림 4에서, EB 60개(60분)까지는 시스템에 대한 부하가 미비하여 성능 인자의 튜닝이 발생하지 않았으며, 시스템 부하가 증가하는 EB 75개(65분) 이상에서는 성능 인자의 튜닝 횟수가 많아졌다. 또한 성능 인자 변경에 의한 튜닝의 성능 효과는 측정 시간 120분이 지나면서 "NO_CHANGE"의 그래프와 각 성능 인자의 그래프 간의 간격이 측정 시간 5분에서의 그래프간격보다 넓음을 통해 확인할 수 있다.

"DB_CACHE_SIZE" 인자는 버퍼 캐시의 크기를 설정하는 것으로 EB 개수의 증가에 따라 캐시되는 자료가 증가하여 이 성능 인자의 적절한 크기 변경이 성능 향상에 영향을 주는 것으로 보인다. "PGA_AGGREGATE_TARGET" 인자는 정렬, 해시 조인 등 메모리를 많이 사용하는 질의에 대한 메모리 크기를 설정하는 것으로 EB 개수가 증가할 때마다 할당되어야 하는 메모리의 크기가 증가하여 해당 성능 인자의 적절한 크기 변경이 성능 향상에 도움을 주는 것으로 판단된다. "SHARED_POOL_SIZE" 인자는 공유 커서, 내장 프로시저, 질의 파싱 등에 사용되는 메모리 영역으로 EB 개수의 증가에 따른 더 많은 질의 파싱 등이 발생하여 좀 더 큰 메모리 할당이 필요하므로 이 성능 인자의 크기 변경이 성능 향상을 가져오고 있다.

각 측정 시간에서 하나 혹은 다수의 웹 상호작용이 응답시간 제한을 초과하지만, 가장 늦은 응답시간으로 빈번하게 튜닝을 유발시키는 웹 상호작용은 "Buy Confirm"과 "Order Display"이다. 그림 5는 "DB_CACHE_SIZE" 인자에 대하여 "Buy Confirm"과 "Order Display"의 응답시간 변화를 보인다. "Buy Confirm"은 측정 시간 35분 이후 1%~57% 정도, "Order Display"는 측정 시간 60분 이후 2%~120% 정도 응답시간을 초과하였다. "Buy Confirm"는 측정 시간 35분, 70분에서, "Order Display"은 측정 시간 60분, 70분에서 응답시간 제한을 초과하여 인자의 크기가 각각 32MB 증가하였으며, 이후 측정 시간에서의 각각의 응답시간은 제한 시간 이내로 감소하였다.

다른 인자들도 이와 유사한 응답시간 변화를 보이고 있다. "PGA_AGGREGATE_TARGET" 인자의 경우, "Buy Confirm"은 5%~200% 정도, "Order Display"은 5%~24% 정도 응답시간을 초과했다. "SHARED_POOL_SIZE" 인자의 경우, "Buy Confirm"은 14%~180% 정도, "Order Display"은 26%~216% 정도 제한된 응답 시간을 초과했다.

시스템 X의 세 가지 성능 인자들은 제시된 두 가지

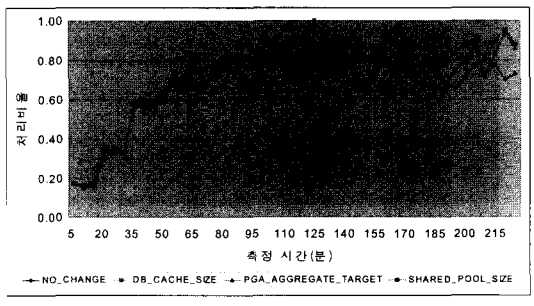

그림 4 처리량 평가 방법에 의한 처리 비율 변화

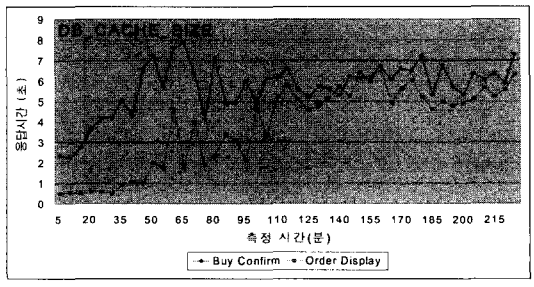

그림 5 응답시간 평가 방법에 의한 응답시간 변화

튜닝 전략을 통해 시스템 성능 향상에 각각 기여하고 있다. 성능 인자 변경을 통해 시스템의 성능 향상을 계획하는 경우, 단일 성능 인자의 변경만을 시도하는 것이 아니라 다중 성능 인자를 복합적으로 변경하는 것이 보다 일반적이다. 따라서 본 논문에서는 다중 성능 인자 변경을 통한 튜닝 전략에 대한 기초 연구 단계로 각 성능 인자들의 단일 성능 평가의 결과를 이용하여 다중 성능 인자의 변경을 시도하고 이를 통해 성능 향상 여부를 알아보았다.

표 4는 다중 성능 인자 변경에 의한 실험 결과를 보인다. EB 개수에 따른 각 성능 인자들의 크기는 동일한 EB 개수에서 수행된 단일 성능 인자 변경을 통한 튜닝 실험에서 사용되었던 성능 인자 크기로 설정하였다. 예를 들어, EB 15개일 때의 WIPS 값을 측정하고자 한다면 처리량 평가 방법의 15개 측정 구간에서 측정된 3가지 성능 인자의 크기 즉, "DB_CACHE_SIZE"는 32MB, "PGA_AGGREGATE_TARGET"은 24MB, "SHARED_POOL_SIZE"는 32MB로 각각 설정하였다. EB가 120개보다 큰 경우에는 3가지 성능 인자의 전체 크기가 실험 대상 시스템의 리소스 용량을 초과함에 따라 EB 개수를 120개까지 로만 제약하여 실험을 수행하였다.

응답시간을 초과하는 웹 상호작용은 "Best Sellers", "Buy Confirm", "Order Display", "Shopping Cart" 들이다. "NO_CHANGE"의 전체 평균 WIPS 값은 4.42이고, 처리량 평가 방법의 전체 평균 WIPS 값은 4.93이고, 응답시간 평가 방법의 전체 평균 WIPS 값은 5.04이다. 이로써 처리량 평가 방법과 응답시간 평가 방법은 "NO_CHANGE" 보다 각각 11.5%, 14% 성능 향상되었다.

처리량 평가 방법의 성능 인자 튜닝 전략을 적용하여 성능 인자를 튜닝하는 경우 대략 6%~14% 정도 성능 향상이 되었으며, 응답시간 평가 방법의 성능 인자 튜닝 전략을 적용하여 성능 인자를 튜닝하는 경우에는 6%~15% 정도 성능 향상이 되었다.

지금까지의 실험 결과로 미루어보아 본 논문에서 제시하는 단일 성능 인자의 튜닝을 위한 튜닝 전략은 적절함을 알 수 있었으며, 단일 성능 인자의 튜닝으로부터 얻어진 개별 성능 인자의 크기를 사용하여 다중 성능 인자를 변경할 경우 시스템의 성능은 11.5%~14% 정도 향상시킬 수 있었다.

3.2 시스템 Y의 결과 및 분석

시스템 Y는 상용 객체-관계형 데이타베이스 시스템이며, 다중 UltraSparc 프로세서, SunOS 5.8에서 운용하였다. 시스템 Y에 대한 실험은 다음과 같이 진행하였다. 첫째, 성능 인자의 변경 판단은 앞서 기술된 시스템 X에서의 성능 인자 변경 판단과 마찬가지로 5분마다 수행하였다. 둘째, 5개의 EB로부터 시작하여 15분마다 새로운 EB를 5개씩 추가하였다. 셋째, 전체 실험 시간은 75분으로 하였다. 즉, 시스템 Y에 대한 실험은 5개의 EB를 바탕으로 실험을 시작하여 15분마다 EB가 5개씩 증가하고, 60분 경과 후에는 EB 개수가 25개가 된다. 성능 인자 변경 판단은 5분마다 수행하여, 총 14번 성능 인자 변경 판단을 수행하게 된다.

시스템 Y에서 실험 대상으로 선정된 성능 인자는 "NUM_DATA_BUFFERS", "SR_BUFFERS", "LOCK_ESCALATION"이다. "NUM_DATA_BUFFERS" 성능 인자는 주 메모리에 캐시되는 데이타 버퍼 개수를 명시하고, "SR_BUFFERS" 성능 인자는 정렬 목적으로 할당되는 버퍼 개수를 명시한다. "LOCK_ESCALATION" 성능 인자는 인스턴스 잠금(instance lock)이 클래스 잠금으로 변경되는 최대 잠금 개수를 명시한다.

시스템 Y에서 각 성능 인자의 변경 크기를 결정하는 실험은 앞에서 기술된 시스템 X의 실험과 동일하며, 실험 결과는 표 5에 나와 있다. "NUM_DATA_BUFFERS"의 기본 크기(P_{init})는 10,000 페이지(80MB), 최대 크기(P_{Max})는 200,000 페이지(1.6GB)이다. "SR_

표 4 다중 성능 인자 변경에 의한 성능 향상

EB	NO_CHANGE (WIPS)	P1 (MB)	P2 (MB)	P3 (MB)	처리량 평가 방법 (WIPS)	P1 (MB)	P2 (MB)	P3 (MB)	응답시간 평가 방법 (WIPS)
15	1.97	32	24	32	1.97	32	24	32	1.97
30	3.81	32	24	32	3.80	32	24	32	3.80
45	4.11	32	24	32	5.12	96	96	64	5.22
60	5.27	64	48	32	5.48	192	168	160	5.78
75	5.22	92	72	96	5.79	256	240	256	5.81
90	5.00	128	96	128	5.91	352	312	352	6.46
105	4.98	192	144	192	5.68	448	384	448	5.65
120	5.01	224	192	224	5.68	544	456	544	5.62

P1: DB_CACHE_SIZE, P2: PGA_AGGREGATE_TARGET, P3: SHARED_POOL_SIZE

표 5 시스템 Y의 성능 인자의 변경 크기

성능 인자	50 %		100 %		200 %	
	변경 크기	처리 비율	변경 크기	처리 비율	변경 크기	처리 비율
NUM_DATA_BUFFERS	5,000 페이지	1.04	10,000 페이지	1	20,000 페이지	0.97
SR_BUFFERS	8 페이지	0.94	16 페이지	1	32 페이지	0.99
LOCK_ESCALATION	50,000	0.99	100,000	1	200,000	0.99

표 6 시스템 Y의 실험 결과

평가 방법	성능 인자	처리 비율	%
NO_CHANGE	각 성능 인자 모두 포함	1	0
처리량 평가 방법	NUM_DATA_BUFFERS	1.0563	+ 5.63
	SR_BUFFERS	1.0536	+ 5.36
	LOCK_ESCALATION	1.0233	+ 2.33
응답시간 평가 방법	NUM_DATA_BUFFERS	1.1182	+ 11.82
	SR_BUFFERS	1.0572	+ 5.72
	LOCK_ESCALATION	1.0478	+ 4.78

BUFFERS"의 기본 크기(P_{init})는 16 페이지(128KB), 최대 크기(P_{Max})는 500 페이지(4MB)이다. "LOCK_ESCALATION"의 기본 크기(P_{init})는 100,000이고 최대 크기에 대한 제한은 없다. 처리 비율은 시스템 X에서의 처리 비율과 동일한 의미를 가진다. 처리 비율이 가장 높은 변경 크기가 최종 변경 크기(P_{inc})로 선정되며, "NUM_DATA_BUFFERS"는 5,000 페이지(40MB), "SR_BUFFERS"는 16 페이지(128KB), "LOCK_ ESCALATION"은 100,000이다.

표 6은 시스템 Y의 성능 인자의 크기 변경에 대한 실험 결과를 보인다. 시스템 Y는 기본 성능 값에 비해 단일 성능 인자의 변경을 수행함으로써 약 2%~11% 정도 성능이 향상되었다.

그림 6은 처리량 평가 방법을 적용한 실험 결과로서 처리 비율의 변화를 보인다. 측정 시간 30분까지는 성능 인자들의 변경을 통한 튜닝이 발생하지 않았으며 처리량에 대한 그래프가 오차 범위 내에서 "NO_CHANGE"의 그래프와 유사하다. 35분~60분까지는 성능 인자 튜닝 전략에 의해 처리량이 "NO_CHANGE"보다 증가하였으며, 60분 이후에는 "LOCK_ESCALATION" 인자의 경우 성능 인자 변경에 의한 튜닝에도 불구하고 오차 범위 내에서 "NO_CHANGE" 보다 다소 낮은 처리 비율을 보인다.

실험 결과를 분석하면, "NUM_DATA_BUFFERS" 인자는 측정 시간 40분, 50분, 65분, 70분에서, "SR_BUFFERS" 인자는 측정 시간 15분, 50분, 65분, 70분에서, "LOCK_ ESCALATION" 인자는 측정 시간 30분, 60분, 65분, 70분에서 처리량이 감소되어 성능 인자 튜닝 전략이 적용되었다. 성능 인자 튜닝 전략의 첫 번째 조건에 의하여 인자가 변경된 경우는 "NUM_DATA_BUFFERS" 인자의 경우 측정 시간 40분,

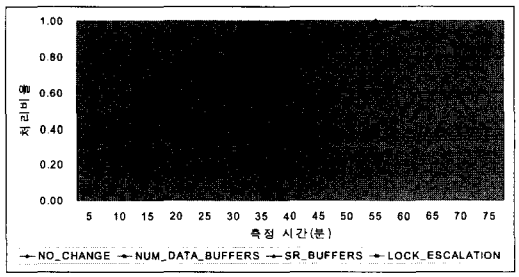

그림 6 처리량 평가 방법에 의한 처리 비율 변화

"SR_BUFFERS" 인자의 경우에는 측정 시간 15분, "LOCK_ESCALATION" 인자의 경우에는 측정 시간 30분인 경우였으며, 그 외의 성능 인자 변경은 성능 인자 튜닝 전략의 두 번째 조건이 적용되었다. "NUM_DATA_BUFFERS" 인자는 측정 시간 40분과 50분의 처리량 감소를 해소하기 위해 인자 크기를 각각 5,000페이지씩 증가하였고, 이후 각각 45분, 55분에서 처리량이 증가되었다. "SR_BUFFERS" 인자는 측정 시간 50분에서 처리량이 감소되어 인자 크기를 16페이지 증가하였고, 이후 55분에서 처리량이 증가되었으며, "LOCK_ESCALATION" 인자는 측정 시간 30분에 인자 크기가 200,000으로 변경되어 55분까지 지속적으로 처리량이 증가되었다.

그림 6에서 EB 10개(30분)까지는 시스템에 대한 부하가 미비하여 성능 인자의 튜닝이 발생하지 않았으며, 시스템 부하가 증가하는 EB 15개(35분) 이상에서는 성능 인자의 튜닝 횟수가 많아지고, 또한 특정 시간 75분에서의 "NO_CHANGE" 그래프와의 간격이 측정 시간 5분에서의 "NO_CHANGE" 그래프와의 간격보다 넓음을 통해 튜닝에 의한 성능 효과를 확인할 수 있었다.

"NUM_DATA_BUFFERS" 인자는 데이타 버퍼 개

수를 설정하는 것으로 EB 개수에 따라 캐시되는 자료도 증가함으로 이 성능 인자의 적절한 버퍼 개수 변경은 성능 향상에 영향을 주는 것으로 보인다. "SR_BUFFERS" 인자는 정렬에 필요한 버퍼 크기를 설정하는 것으로 증가되는 EB마다 할당되어야 하는 버퍼의 크기가 증가하게 되어 적절한 버퍼 크기 변경은 성능 향상에 도움을 주는 것으로 판단된다. "LOCK_ESCALATION" 인자는 인스턴스의 잠금이 클래스 잠금으로 변경되는 최대 잠금 개수를 설정하는 인자로 증가되는 EB 개수에 따라 잠금의 대상이 되는 자료가 증가하고, 이로 인해 인스턴스 잠금에서 클래스 잠금으로 변경이 필요한 자료도 증가하게 되어 해당 성능 인자의 변경이 성능 향상에 영향을 주고 있는 것으로 보인다.

각 측정 시간에서 하나 혹은 다수의 웹 상호작용이 응답시간 제한을 초과하지만, 가장 늦은 응답시간으로 빈번하게 튜닝을 유발시키는 웹 상호작용은 "Best Sellers"와 "Buy Confirm"이다. 그림 7은 "NUM_DATA_BUFFERS" 인자에 대하여 웹 상호작용 "Best Sellers"와 "Buy Confirm"의 응답시간 변화를 보인다. "NUM_DATA_BUFFERS" 인자의 경우, "Best Sellers"의 응답시간은 모든 측정 시간에서 최저 1%에서 최대 약 490% 정도 초과했으며, "Buy Confirm"의 응답시간은 측정 시간 45분 이후 약 9%~270% 정도 초과했다. 각 웹 상호작용의 응답시간이 제한 시간을 초과하면, "NUM_DATA_BUFFERS" 인자는 5,000페이지씩 크기가 증가된다. "Best Sellers"는 측정 시간 5분, 20분, 50분에서, "Buy Confirm"은 측정 시간 30분에서 응답시간 제한을 초과하여 인자의 크기를 각각 5,000 페이지 증가시켰다. 이후 각각의 측정 시간에서는 응답시간이 다소 감소하였으며, "Buy Confirm"은 측정 시간 35분, 40분에서 응답시간이 제한 시간 이내였다.

다른 인자들도 이와 유사한 응답시간 변화를 보이고 있다. "SR_BUFFERS" 인자의 경우, "Best Sellers"의 응답시간은 27%~880% 정도 초과했으며, "Buy Confirm"의 응답시간은 10%~790% 정도 초과했다. "LOCK_ESCALATION" 인자의 경우, "Best Sellers"의 응답시간은 3%~895% 정도 응답시간을 초과했으며, "Buy Confirm"의 응답시간은 약 55%~780% 초과했다.

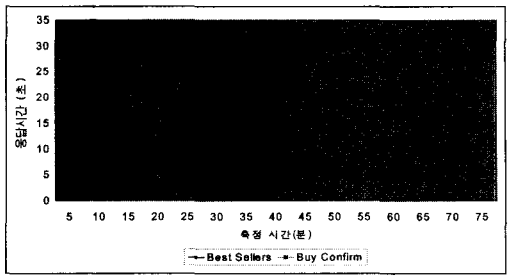

그림 7 응답시간 평가 방법에 의한 응답시간 변화

시스템 Y의 세 가지 성능 인자들은 제시된 두 가지 튜닝 전략을 통해 시스템 성능 향상에 각각 기여하고 있다. 시스템 Y도 시스템 X에서와 같이 다중 성능 인자를 복합적으로 변경하는 경우의 성능 향상을 확인해 보기 위해 앞선 실험을 통해 얻어진 성능 인자의 크기를 적용하여, EB 개수별로 TPC-W 실험을 수행하였다. 실험 방법은 시스템 X에서의 실험 방법과 동일하다. 표 7은 실험 결과를 보인다. "NO_CHANGE"의 평균 WIPS 값은 1.012이고, 처리량 평가 방법의 평균 WIPS 값은 1.184이며, 응답시간 평가 방법의 평균 WIPS 값은 1.180이다. 이로써 처리량 평가 방법과 응답시간 평가 방법은 각각 "NO_CHANGE"보다 17%, 16% 성능 향상되었다.

처리량 평가 방법의 성능 인자 튜닝 전략을 적용하여 튜닝하는 경우 대략 각 성능 인자의 기본값을 사용하는 "NO_CHANGE"보다 2%~5% 정도 성능 향상이 되었으며, 응답시간 평가 방법의 성능 인자 튜닝 전략을 적용하여 튜닝하는 경우에는 "NO_CHANGE"보다 4%~11% 정도 성능 향상이 되었다.

앞서 제시된 실험 결과 및 분석 결과를 미루어보아 본 논문에서 제시하는 단일 성능 인자의 튜닝을 위한 튜닝 전략은 시스템 Y에서도 적절함을 알 수 있었으며, 단일 성능 인자의 튜닝으로부터 얻어진 성능 인자의 크기를 활용한 다중 성능 인자 변경을 수행할 경우 시스

표 7 다중 성능 인자 변경에 의한 성능 향상

EB	NO_CHANGE (WIPS)	P1 (페이지)	P2 (페이지)	P3 (개)	처리량 평가방법 (WIPS)	P1 (페이지)	P2 (페이지)	P3 (개)	응답시간 평가방법 (WIPS)
5	0.59	10000	32	100000	0.65	25000	16	300000	0.61
10	0.82	10000	32	200000	0.90	40000	64	600000	0.82
15	0.95	15000	32	200000	1.38	55000	112	900000	1.44
20	1.37	20000	48	300000	1.54	70000	160	1200000	1.54
25	1.32	35000	96	600000	1.49	85000	208	1500000	1.49

P1: NUM_BUFFERS, P2: SR_BUFFERS, P3: LOCK_ESCALATION

템의 성능은 "NO_CHANGE"보다 16%~17% 정도 향상시킬 수 있었다.

4. 결론 및 향후 계획

본 논문은 부하에 따른 성능 인자의 크기를 적절하게 튜닝하기 위해 TPC-W 환경에서 데이타베이스 시스템의 성능 인자에 대하여 처리량 평가 방법과 응답시간 평가 방법의 튜닝 전략을 제시하였다. 제시된 성능 인자 튜닝 전략에 대하여 상용 데이타베이스 시스템을 사용하여 실험을 수행하였으며, 그 결과 시스템 X와 시스템 Y의 각 성능 인자들은 각각 6%~15%, 2%~11% 정도 성능 향상이 되었다. 따라서 본 논문에서 제시된 튜닝 전략은 데이타베이스 시스템의 성능 향상에 기여할 수 있는 적당한 튜닝 전략임을 알 수 있다.

성능 향상에 가장 크게 영향을 미치는 성능 인자는 시스템 X에서는 "DB_CACHE_SIZE" 인자이고, 시스템 Y에서는 "NUM_DATA_BUFFERS" 인자로서 두 성능 인자 모두 데이타 버퍼 관련 인자들이다. 처리량 평가 방법의 튜닝 전략에서 측정 시간 초기에는 작업부하가 미비하여 성능 인자 변경에 의한 튜닝이 거의 발생하지 않았지만, 작업부하가 증가하면서 성능 인자의 변경을 통한 튜닝이 발생하여 "NO_CHANGE"보다 성능이 향상되었으며, 개별 성능 인자들의 특성에 따라 그 차이는 다양하게 발생하였다. 응답시간 평가 방법의 튜닝 전략에서 작업부하가 증가하면서 응답시간이 제한 시간을 초과하여 성능 인자 변경에 의한 튜닝이 발생하였으며, 그 결과 응답시간이 부분적으로 감소함을 확인할 수 있었다.

데이타베이스 시스템의 성능 인자 튜닝은 일반적으로 다중 성능 인자를 튜닝하는 것이다. 향후 본 논문의 단일 성능 인자를 통한 튜닝 및 기초적인 다중 성능 인자 튜닝 결과를 기반으로 적절한 모델링 기법을 사용한 다중 성능 인자의 튜닝 전략 및 기법에 대한 연구가 요구된다.

참 고 문 헌

[1] D. Shasha and P. Bonnet, "Database Tuning: Principles, Experiments, and Troubleshooting Techniques," p. 415, Morgan Kaufmann Publishers, 2002.

[2] D. Shasha, "Tuning Databases for High Performance," ACM Computing Surveys, Vol. 28, No. 1, pp. 113-115, 1996.

[3] K. D. Ahn, J. S. Oh, and S. H. Lee, "Development of a Tuning Aid for Database Systems," The Transactions of the Korea Information Processing Society, pp. 3311-3322, 2000.

[4] Oracle Corporation, "Designing and Tuning for Performance," http://download-west.oracle.com/docs/cd/A87860_01/doc/server.817/a76992.pdf, 1999.

[5] S. Chaudhuri and R. Narasayya, "AutoAdmin 'What-if' Index Analysis Utility," Proceedings of ACM SIGMOD Conference, pp. 367-378, 1998.

[6] D. Menasce, D. Barbara, and R. Dodge, "Preserving QoS of E-commerce Sites Through Self-Tuning: A Performance Model Approach," Proceedings of ACM Conference on Electronic Commerce, pp. 224-234, 2001.

[7] Transaction Processing Performance Council, http://www.tpc.org/.

[8] W. D. Smith, "TPC-W: Benchmarking an Ecommerce Solution," http://www.tpc.org/tpcw/TPC-W_Wh.pdf.

[9] H. S. Lee, S. J. Kim, and S. H. Lee, "Parameter Tuning in Database Systems," in preparation, 2003.

[10] D. DeWitt, "The Wisconsin Benchmark: Past, Present, and Future, In: The Benchmark Handbook," pp. 269-315, Morgan Kaufmann Publishers, 1993.

[11] The BEA Systems, http://www.bea.com/.

류 문 수

1999년 부경대학교 컴퓨터공학과 졸업(학사). 2003년 숭실대학교 대학원 컴퓨터학과 졸업(석사). 1999년~2001년 (주)필로소프트, 연구원. 2003년~현재 (주)핸디소프트, 책임연구원. 관심분야는 데이타베이스 시스템 성능 평가 및 튜닝

정 희 진

1993년 우석대학교 전산학과(학사). 1995년 숭실대학교 대학원 컴퓨터학과(석사) 1995년~2000년 (주)핸디소프트 기술연구소, 선임연구원. 2002년~2003년 숭실대학교 정보미디어 연구소, 전임연구원 2000년~현재 숭실대학교 대학원 컴퓨터학과 박사과정. 관심분야는 데이타베이스 시스템 성능 평가 및 튜닝, XML 데이타베이스

이 상 호

1984년 서울대학교 전산공학과 졸업(학사). 1986년 미국 노스웨스턴대 전산학과(석사). 1989년 미국 노스웨스턴대 전산학과(박사). 1990년~1992년 한국전자통신연구원, 선임연구원. 1999년~2000년 미국 조지메이슨대, 소프트웨어정보공학과, 교환 교수. 1992년~현재 숭실대학교 컴퓨터학부 교수 관심분야는 인터넷 데이타베이스, 데이타베이스 시스템 성능 평가 및 튜닝