데이터베이스 시스템 벤치마크를 위한 실세계 부하 생성 도구

김 기 욱[†] · 정 회 진[†] · 이 상 호[††]

요 약

데이터베이스 시스템 벤치마크는 결과 값의 극대화를 위해 실험 대상 시스템의 가용 자원을 최대화한 상태에서 수행되어 실세계 환경과는 괴리된다는 한계성을 가지고 있다. 실세계에서의 작업 환경과 유사한 환경에서의 벤치마크 시험을 위해 본 논문에서는 기존 벤치마크를 보완할 수 있는 실세계 부하 생성 도구를 제안한다. 본 부하 생성 도구는 시스템의 메모리, 디스크, CPU를 활용하여 운영체제에 직접적인 부하를 생성하며, 실세계 부하와 유사한 통합 부하 생성을 지원한다. 본 논문에서는 각 부하 생성 방식, 개발된 부하 생성 도구의 구조, 특징, 구현 방법 등을 기술하였다. 또한 위스콘신 벤치마크를 TPC-C 벤치마크 및 부하 생성 도구와 함께 수행하여 두 성능 평가 실험 결과를 비교하고, 이를 통해 제안하는 부하 생성 도구의 적절성을 보였다.

A Real-World Workload Generation Tool for Database System Benchmarks

Kee Wuk Kim[†] · Hoe Jin Jeong[†] · Sang Ho Lee[††]

ABSTRACT

Database system benchmarks, which are usually evaluated to use the maximized resource in order to get the best results, are not likely to simulate the real environment. We propose a workload generator that helps benchmarks be executed in the environment similar to a real world. The workload generator can create memory-bound, CPU-bound, and I/O-bound workloads. The workload generator allows users to create an integrated workload, which is similar to a real workload users run across in practice. Finally, we conducted the experiments that the Wisconsin benchmark was performed with the TPC-C and with the workload generation tool, and showed the feasibility of the proposed workload generation tool by comparing with two experimental results.

키워드 : 부하 생성 도구(Workload Generation Tool), 부하 생성(Workload Generation), 실세계 부하(Real Workloads), 벤치마크(Benchmark)

1. 서 론

현대 사회에 응용되고 있는 모든 시스템 및 소프트웨어는 개발 단계에서부터 사용자의 선택에 이르기까지 많은 평가 실험을 필요로 한다. 데이터베이스 시스템 또한 실세계의 업무 영역을 잘 지원하고 있는지 평가할 필요가 있으며, 이를 위해 데이터베이스 시스템 벤치마크가 실세계 응용 프로그램의 부하나 특성을 바탕으로 설계 및 개발되었다[1]. 데이터베이스 시스템 벤치마크의 대상 분야는 크게 OLTP(online transaction processing) 분야와 DSS(decision support systems) 분야, 웹 전자 상거래(e-commerce) 분야로 나눌 수 있다[2]. OLTP 분야에는 TPC-C 벤치마크[3]가 대표적이고, DSS 분야에는 BORD(benchmark for object-relational databases)[4]와 SetQuery 벤치마크[5], Wisconsin 벤치마크[6] 등이 있으며, 전자 상거래 분야에는 TPC-W 벤치마크가 있다.

앞서 언급된 데이터베이스 시스템 벤치마크들은 각 분야의 핵심 부하(workload) 부분만을 모델링 하고 있으며, 이로 인해 실세계 부하 중 핵심적이지 않은 부하 부분들은 데이터베이스 시스템 벤치마크에도 반영되지 않는다는 문제점을 가진다. 일반적으로 데이터베이스 시스템 벤치마크 시험은 결과 값의 극대화를 위해 실험 대상 시스템의 불필요한 응용 프로그램을 종료하고 가용 자원을 최대화한 상태에서 수행된다. 하지만, 이러한 실험 환경은 다양한 부하와 함께 해당 분야의 응용 프로그램이 수행되는 실세계 환경을 사실적으로 반영하지 못하고 있다. 결국 비현실적인 수행 환경에서 측정된 벤치

※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌습니다.
† 준 회 원 : 숭실대학교 대학원 컴퓨터학과
†† 정 회 원 : 숭실대학교 컴퓨터학과 교수
논문접수 : 2004년 6월 16일, 심사완료 : 2004년 11월 15일

마크 결과는 사용자들에게 올바른 판단 기준이 되지 못한다.

데이터베이스 시스템 벤치마크는 신뢰할 수 있는 실험 결과를 위해 실세계의 해당 분야를 대표할 수 있는 부하를 사용하여야 하지만, 기존 데이터베이스 시스템 벤치마크들은 앞서 언급된 한계성과 문제점을 지니고 있다. 이런 한계성을 극복하기 위해서는 실세계에서의 작업 환경과 유사한 환경을 만들어 벤치마크를 수행토록 하는 것이 필요하다. 이를 위해서 기존 데이터베이스 시스템 벤치마크를 보완할 수 있는 부하 생성 도구가 필요하다.

실세계 데이터베이스 환경에서는 발생 할 수 있는 모든 상황에 적절하게 대처하기 위해 데이터베이스 시스템의 성능에 대한 예측이나, 부하의 크기에 따른 성능 변화 예측이 필요하다. 이를 위해서 현재 데이터베이스 시스템의 성능을 알아야 하며, 부하의 크기 변화를 위해 일정 크기로 부하를 증가시키거나 감소시켜야 한다. 본 논문에서 제안하는 부하 생성 도구는 부하의 크기를 조절할 수 있어 부하의 영향을 실제 데이터베이스 시스템 환경에서 미리 파악할 수 있고, 또한 사용자가 원하는 때에 부하 제어가 가능하여 매번 비슷한 상황을 만들어 반복 실험을 가능케 한다.

부하와 관련하여 지금까지의 연구는 부하 생성 도구의 개발과 같은 직접적인 분야에서 보다는 부하 모델링과 관련된 기초적인 연구가 진행되었다. [7]에서는 부하 모델링의 정의를 통계적 방법에 기초하여 기술하였고 다양한 통계적 모델 구축 방법을 고려하여 구조적 모델링에 대한 개념을 정리하였다. [8]은 데이터의 유사성 분석을 위한 대표적인 방법인 클러스터링 알고리즘과 부하의 동적인 작용을 나타내기 위해 사용되는 그래프 방법을 이용하여 부하의 정적인 면과 동적인 면을 분류하였고 이를 바탕으로 통계적 분석을 시도하였다. 통계적 분석이 완료되면 여러 개의 측정 요소를 바탕으로 부하 모델에 대해 대표성을 평가하였다.

본 논문은 데이터베이스 시스템 벤치마크를 위한 부하 생성 도구를 기술한다. 부하 생성 도구는 실세계 부하와 유사하고, 사용자들이 쉽게 조작 가능한 통합 부하 생성을 지원한다. 또한, 다른 프로세스의 영향을 받지 않고 독립적으로 실행되도록 설계되어 사용자가 원하는 다양한 형태의 부하를 생성할 수 있다. 사용자는 부하 생성 도구에서 제공되는 다양한 파라미터를 사용하여 주어진 환경에서의 응용프로그램에 대한 부하 영향을 관찰할 수 있다.

본문에서는 부하 생성 도구의 정확한 부하 생성 여부를 측정하기 위한 실험을 수행하여 부하 생성 도구의 입력 파라미터 값으로 다양한 값을 입력하였을 때 원하는 부하를 정확하게 생성할 수 있는지를 알아본다. 또한, TPC-C 벤치마크 수행 후 측정된 부하와 같은 크기의 부하를 생성할 수 있는지를 알아보고, 부하 생성 도구의 전반적인 성능을 알아보기 위해 Wisconsin 벤치마크와의 연동 실험을 수행한다.

부하 생성 도구의 부하 생성 방식은 운영체제에 직접 부하를 주는 방식으로서, 메모리 위주(memory-bound) 부하 생성, 입출력 위주(I/O-bound) 부하 생성, CPU 위주(CPU-bound) 부하 생성으로 나뉜다. 운영체제에서 수행되는 모든 프로그램은 일반적으로 메모리 관리, 입출력 처리, CPU 계산 작업 중 하나의 작업이 주가 되어 수행되기 때문에 부하를 세분화하여 나눌 수 있다[9].

본 논문의 구성은 다음과 같다. 2장에서는 부하 생성 도구의 모듈 구성과 각 부하 생성 모듈에 대해 설명한다. 또한, 각 부하 생성 모듈에서의 부하 정의와 부하 생성 방식에 대해 설명한다. 3장에서는 각 부하 생성 방식의 적절성을 알아보기 위해 부하 생성 도구의 각 모듈별 실험 결과 및 분석 내용을 기술한다. 4장에서는 부하 생성 도구의 통합 모듈과 데이터베이스 시스템 벤치마크를 연동하여 실험한 적용 사례를 설명한다. 5장에서는 결론과 향후 연구 방향에 대하여 기술한다.

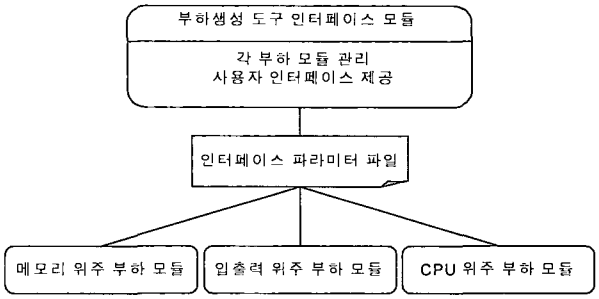
2. 부하 생성 도구

부하 생성 도구를 통해 생성 가능한 부하의 종류는 메모리 위주 부하, 입출력 위주 부하, CPU 위주 부하로 나뉘어 진다. 부하 생성 도구는 다중 프로세스 환경을 지원하는 유닉스(UNIX) 계열의 중, 소형 시스템에서 운영 체제 자원을 직접 이용하여 사용자가 입력한 값에 의거하여 부하를 생성한다. 운영 체제가 제어하는 자원을 사용자가 정확하게 제어하는 것은 불가능하며, 생성되는 부하의 크기가 사용자가 입력한 부하의 크기를 기준으로 일정 오차 범위 내에서 유지되도록 부하 생성 작업이 진행된다.

부하 생성 도구를 활용한 부하 생성 방법은 두 가지로 나뉜다. 하나는 사용자가 임의의 값을 입력하여 부하를 생성하는 경우이고, 다른 하나는 특정 형태의 부하를 유사하게 생성하는 경우이다. 특정 형태의 부하를 발생시키기 위해서는 해당 부하에 대한 여러 통계 정보가 필요하다. 통계 정보를 추출해 내기 위해 운영체제의 "top" 명령어와 "iostat" 명령어를 사용한다. "top" 명령어는 프로세스 단위의 자원 사용량과 시스템 전체의 자원 사용량을 사용자가 쉽게 알 수 있도록 여러 가지 통계 항목들을 제공한다. "iostat" 명령어를 통해서는 KB/s(kilo bytes per second) 단위로 제공되는 실제 메모리와 스왑 메모리(swap memory) 사이의 데이터 교환 크기 및 디스크의 입출력 양을 알 수 있다.

2.1 부하 생성 도구 모듈 구성

부하 생성 도구의 모듈 구성은 (그림 1)과 같다. 메모리 위주 부하 모듈과 CPU 위주 부하 모듈은 시작 시간, 수행 시간, 부하 크기를 인터페이스 모듈로부터 전달받아 부하를 생성한다. 입출력 위주 부하 모듈은 디렉터리 개수와 디렉터리 경로를 모듈로부터 추가로 전달받아 부하를 생성한다. 부하 생성 도구의 세 가지 모듈은 개별적으로 실행될 수도 있고 서로 조합하여 실행될 수도 있다. 인터페이스 파라미터 파일에는 앞서 언급된 항목 등 부하 생성에 필요한 내용이 사용자로부터 입력되어 기록된다. 부하 생성 도구 인터페이스 모듈은 각 모듈들을 통합하여 관리하고, 사용자 인터페이스 처리를 담당한다.



(그림 1) 부하 생성 도구의 모듈 구성

메모리 위주 부하 모듈은 스왑 메모리와 실제 메모리 사이에 데이터 치환이 발생하도록 유도하여 부하를 생성한다. 입출력 위주 부하 모듈은 디스크 상에서 데이터의 복사 및 삭제 작업을 통해 부하를 생성한다. CPU 위주 부하 모듈은 단순 연산 작업을 수행하여 부하를 생성한다.

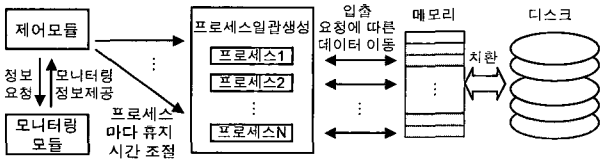
인터페이스 파라미터 파일에는 모듈들이 위치한 기본 디렉터리 경로, 입출력 위주 부하 생성 시 사용할 디렉터리 개수와 절대 경로, 각 부하 크기와 부하 생성 시간 등이 저장된다. 부하 생성 시간은 부하 생성 도구가 수행될 총 시간을 말하며, 분 단위를 사용한다. 부하 크기는 부하 표시기(workload indicator)를 사용하여 입력한다. 부하 표시기란 각 모듈을 통해 생성할 부하 크기를 입력받는 엔트리(entry)를 의미한다.

부하 생성 도구 인터페이스 모듈은 각 부하 생성 모듈에 인자 값으로 부하 생성 시작 시간과 인터페이스 파라미터 파일을 전달해준다. 인자 값을 전달받은 각 모듈들은 인터페이스 파라미터 파일의 내용을 기반으로 부하 생성 작업을 수행하며, 일정 시간마다 수행 시간을 계산하여 지정 시간 동안만 부하 생성 작업을 수행한다.

2.2 메모리 위주 부하 모듈

메모리 위주 부하는 사용자 프로세스가 실제 메모리 공간 중에서 가용 메모리 공간 보다 많은 양의 메모리를 사용함으로 인해 운영 체제가 메모리를 관리하게 하는 일을 말한다. 스왑 메모리와 실제 메모리는 상호간에 치환(swapping)을 쉽게 하기 위해 페이지라는 작은 조각으로 나뉜다. 운영 체제는 프로세스가 스왑 메모리에 있는 페이지를 실제 메모리로 가져올 때, 실제 메모리에 빈 공간이 없다면 LRU (least recently used) 알고리즘에 따라 실제 메모리에서 제거될 페이지를 찾아 빈 공간을 만든다. (그림 2)에서 제어 모듈은 모니터링 모듈로부터 전달받은 통계 정보를 기반으로 일괄 생성된 프로세스의 휴지 시간(sleep time)을 조절한다. 메모리 위주 부하는 2개 이상의 프로세스가 실제 메모리 공간을 대상으로 경합할 때 운영 체제가 치환 작업을 통해 실제 메모리 상에 빈 공간을 만드는 과정에서 생성된다.

본 논문에서는 메모리 위주 부하의 생성을 위해 실제 메모리 크기의 1.5배 크기에 해당하는 데이터를 사용한다. 데이터의 크기가 실제 메모리 크기 이하인 경우에는 프로세스 간 경합이 발생하지 않았고, 실제 메모리의 1.5배 이상 되는 크기의 데이터를 사용한 경우에는 데이터의 크기 변화에 관계없이 사용자가 설정한 부하 크기에 따라 메모리 위주 부하의 발생 양이 동일하였다.



(그림 2) 메모리 위주 부하

메모리 위주 부하의 생성을 위해 데이터 생성 작업과 생성되는 데이터를 메모리에 적재하는 작업이 수행된다. 실제 메모리에 데이터가 다 적재된 후에는 나머지 데이터를 메모리에 적재하기 위해 운영 체제가 실제 메모리 공간의 일부를 스왑 메모리와 치환한다. 데이터 적재 과정이 완료되면 운영 체제로 하여금 실제 메모리와 스왑 메모리 간에 치환 작업을 수행하도록 실제 메모리에 적재된 일정 크기의 데이터를 읽어와 차례대로 참/거짓 비교를 수행한다. 이때 생성된 부하 크기와 사용자가 입력한 부하 값을 비교하여 약 ±5% 이내의 오차 범위를 갖도록 한번에 읽어올 수 있는 데이터의 크기를 설정한다. 데이터의 크기는 200KB를 기본 값으로 한다. 생성된 부하 크기가 사용자가 입력한 부하 값보다 크면 한번에 읽어올 데이터의 크기를 줄이고, 작으면 데이터의 크기를 늘린다. 조절 방법은 측정된 부하 크기에 따라 100KB 단위로 증감하여 부하 크기를 측정하고 그 다음에는 10KB 단위, 1KB 단위 순으로 조절한다.

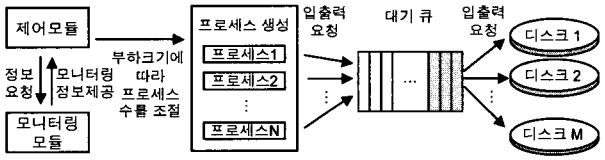
메모리 위주 부하의 생성량은 사용자가 입력한 값에 기반을 둔다. 하지만, 운영 체제가 제어하는 자원을 사용자가 사용하여 정확하게 제어하는 것은 어렵다. 따라서 메모리 위주 부하 생성 프로세스는 적절한 부하 조절을 위해 스왑 메모리로부터 일정 크기의 데이터를 읽어올 때마다 휴지 시간을 갖는다. 각각의 프로세스들이 짧은 휴지 시간을 가지면서 스왑 메모리 상의 데이터를 읽어 들이면 생성하는 부하 크기는 증가하고, 긴 휴지 시간을 갖는 경우에는 부하의 크기는 감소한다. 생성되는 메모리 위주 부하의 크기에 따라 조절할 수 있는 최소 값인 1초 단위로 휴지 시간을 증가시키거나 감소시키면서 부하의 크기를 조절한다. 휴지 시간은 프로세스마다 개별적으로 조절된다.

메모리 위주 부하의 크기는 "top" 명령어의 여러 통계 정보 값 중 "io wait" 항목 값과 "iostat" 명령어를 통해 알 수 있는 실제 메모리와 스왑 메모리 사이의 데이터 교환 크기를 사용하여 알 수 있다. 실제 메모리와 스왑 메모리 사이의 데이터 교환 크기는 실제 메모리로의 스왑 인(swap in)된 데이터의 크기와 스왑 아웃(swap out)된 데이터의 크기를 나타낸다. 메모리 위주 부하의 크기는 사용자가 쉽게 이해할 수 있

도록 "%" 단위를 사용하는 "io wait" 항목 값으로 나타낸다. 메모리 위주 부하의 크기가 사용자가 입력한 수치에 도달했는지의 여부는 "io wait" 항목 3회 값의 산술 평균값을 사용자가 입력한 수치와 비교했을 때 5회 연속 오차 범위 내에 있는지 판단하여 결정한다.

2.3 입출력 위주 부하 모듈

입출력 위주 부하는 사용자 프로세스가 디스크에 대해 많은 입출력 작업을 요청하여 운영 체제가 디스크 입출력 요청을 처리하게 하는 일을 말한다. 사용자 프로세스로부터 발생된 입출력 요청은 대기 큐(wait queue)를 거쳐 디스크로 전달된다. (그림 3)에서 제어 모듈은 모니터링 모듈로부터 전달받은 통계 정보 값에 따라 입출력 위주 부하 프로세스의 수를 조절한다. 입출력 위주 부하는 하나 이상의 프로세스가 하나 이상의 디스크에 대해 입출력 작업을 요청할 때 생성된다.



(그림 3) 입출력 위주 부하

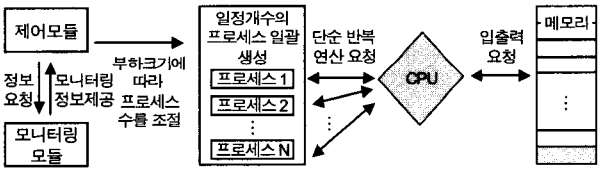
입출력 위주 부하는 디스크에서 일정 크기의 데이터 파일들을 복사하고 삭제하는 작업을 통해 생성되며, 그 크기는 데이터 파일의 복사와 삭제 작업을 수행하는 프로세스의 수와 데이터 파일의 크기를 변경하여 조절한다. 즉, 복사와 삭제 작업을 수행하는 프로세스의 수를 늘리거나 데이터 파일의 크기를 크게 하면 운영 체제가 많은 입출력 요청을 처리하게 되므로 입출력 위주 부하의 크기는 증가하게 된다. 입출력 위주 부하의 크기는 "%" 단위를 사용하는 "io wait" 항목 값으로 나타난다. 입출력 위주 부하를 통해 생성되는 부하 값이 사용자가 입력한 부하 값에 도달하는 시간을 줄이기 위해 크기가 다른 4개의 데이터 파일을 사용하며 각 데이터 파일마다 발생시킬 수 있는 부하의 크기가 다르다. 생성할 데이터 파일들 중 가장 작은 데이터 파일의 크기는 파일의 복사 및 삭제 작업을 반복 수행 하였을 때 "io wait" 항목 값이 약 3%가 되도록 설정하며 나머지 데이터 파일의 크기는 각각 약 5%, 6%, 7%가 되도록 설정한다. 사용자가 입력한 부하의 크기에 따라 프로세스에서 사용될 파일의 크기는 동적으로 설정한다. 입출력 위주 부하의 크기 측정 시에는 "io wait" 통계 정보 3회 값에 대한 산술 평균 값을 구해 반영한다.

입출력 위주 부하 생성은 부하 생성 시 필요한 데이터 파일들을 특정 디렉터리에 생성하는 작업으로부터 시작한다. 이후 작업 프로세스 1개를 생성하고, 해당 프로세스가 미리 생성한 데이터 파일들 중 가장 큰 크기의 데이터 파일을 사용자로부터 입력받은 디렉터리 경로 상에서 복사 및 삭제 작업을 수행한다. 측정된 입출력 부하의 크기가 사용자가 입력한 부하의 크기보다 작은 경우에는 프로세스 1개를 새로 생성하여 같은 크기의 데이터 파일에 대해 복사 및 삭제 작업을 수행하고, 측정된 부하의 크기가 사용자가 입력한 부하의 크기보다 큰 경우에는 직전에 사용된 데이터 파일보다 작은 크기의 데이터 파일을 사용하여 복사 및 삭제 작업을 새로 수행한다. 가장 작은 데이터 파일을 통해 생성되는 부하 크기가 약 3%가 되므로 측정된 부하의 크기와 사용자가 입력한 부하 크기 사이에는 약 ±3%의 오차 범위가 생길 수 있다.

입출력 위주 부하 생성 작업은 파일 복사 작업, 삭제 작업, 휴지 시간으로 구성된다. 입출력 위주 부하를 생성할 때, 데이터 파일의 크기에 따라 입출력 부하 외에 CPU 작업 부하가 추가로 생성된다. 그러나 파일의 복사 및 삭제 시간 외에 휴지 시간을 갖게 되면 생성되는 CPU 부하의 크기를 감소시켜 사용자가 생성하고자 하는 입출력 부하만을 생성할 수 있게 된다.

2.4 CPU 위주 부하 모듈

CPU 위주 부하는 사용자 프로세스가 CPU로 하여금 많은 계산을 수행하게 하는 일을 말한다. (그림 4)에서 제어 모듈은 CPU 위주 부하를 발생시킬 프로세스를 일괄 생성한 후 모니터링 모듈로부터 전달받은 통계 정보 값에 따라 프로세스의 개수를 조절한다. CPU 위주 부하는 여러 개의 프로세스가 CPU로 하여금 계산을 수행하도록 연산 요청을 할 때 생성된다.



(그림 4) CPU 위주 부하

CPU 위주 부하를 만들기 위해 하나의 프로세스는 단순 증가 연산과 휴지를 단위 작업으로 수행한다. 단순 증가 연산 작업은 식 (1)에서 계산된 횟수만큼 반복 수행된다.

CPU 개수 * CPU 클락 주파수 * 시스템 클락 주파수 * 보정계수 (1)

식 (1)에 사용된 CPU 개수, CPU 클락 주파수, 시스템 클락 주파수는 운영 체제에서 제공하는 "prtdiag" 명령어를 통해 구한다. 보정계수 값은 많은 실험을 통해 사용자가 1% 단위로 입력한 CPU 위주 부하의 크기와 측정된 CPU 위주 부하 크기의 오차를 줄이기 위한 경험 값이다. 보정계수 값은 1.0부터 0.5씩 증가시키면서 하나의 프로세스가 1%의 CPU 부하를 생성할 때까지 측정하여 구한다. 보정계수 값을 0.5 증가시켰을 때 하나의 프로세스가 1% 부하 크기를 초과하여 부하를 생성하고, 0.5를 감소시켰을 때 1% 미만의 부하가 생성된다면 0.1씩 증가시킨다. 보정계수 값을 0.1 증가시켰을 때 프로세스

하나가 1%를 초과한 부하를 생성하고, 0.1을 감소시켰을 때 1% 미만의 부하가 생성된다면 0.05씩 증가시킨다. 이와 같은 방법을 반복 수행하여 하나의 프로세스가 1%에서 ±0.01%의 범위를 갖는 부하를 생성하도록 하는 보정계수 값을 구한다.

운영체제는 사용자가 입력한 CPU 위주 부하 크기에 따라 부하 생성 작업을 수행할 프로세스를 N개까지 생성한다. 작업 프로세스는 사용자가 입력한 부하의 크기가 10% 이상일 경우에는 일정 개수까지 일괄 생성되며, 10% 미만일 경우에는 하나씩 생성된다. 일괄 생성되는 프로세스의 개수는 사용자가 입력한 부하 크기에서 10을 뺀 수만큼 생성된다. 작업 프로세스의 개수는 작업을 수행한 후 측정치를 구하고 측정치가 사용자가 입력한 부하의 크기보다 크거나 작은 경우에는 프로세스를 1개씩 감소시키거나 증가시키면서 보정한다.

CPU 위주 부하의 크기는 "top" 명령어의 수행을 통해 알 수 있는 통계 정보에서 사용자 CPU 사용량을 나타내는 "user" 항목 값과 운영체제 커널의 CPU 사용량을 나타내는 "sys" 항목 값의 합으로 측정한다. 3회 측정치에 대해 산술평균을 구한 값은 다음 부하 생성에 반영된다. CPU 위주 부하를 생성한 후 측정된 값과 사용자가 입력한 CPU 위주 부하 목표 값 사이에는 약 ±1%의 오차 범위가 생길 수 있다.

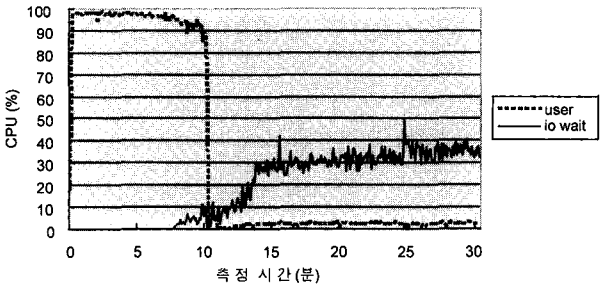
3. 각 모듈 별 실험 결과 및 분석

본 장에서는 각 부하 생성 방식의 적절성을 알아보기 위해 모듈 별로 실험한 결과 및 분석 내용을 기술한다. 실험 결과 값은 메모리 위주 부하 모듈, 입출력 위주 부하 모듈, CPU 위주 부하 모듈을 통해 일정 시간동안 부하를 생성하도록 설정한 후 통계 정보 명령어를 사용하여 측정한다. 실험에 사용된 장비는 Sun사의 중형 서버인 Enterprise E3500이다. E3500 서버는 운영 체제로 SunOS 8을 사용하며, CPU는 400MHz 2개, 메모리는 2GB(giga bytes)를 가진다. 하드디스크는 8개가 설치되어 총 160GB의 크기를 가진다.

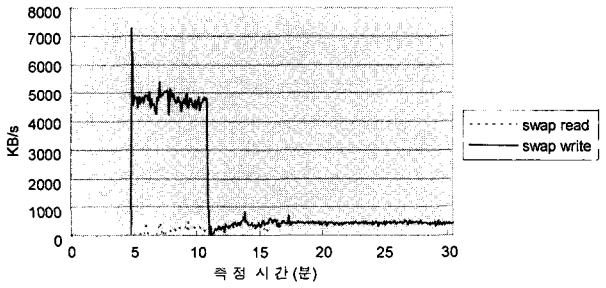
3.1 메모리 위주 부하 생성 실험

(그림 5)는 부하 생성을 위한 입력 값으로 여러 가지 수치를 입력해 수행한 실험 결과들 중 하나이며, 사용자가 입력한 메모리 위주 부하의 생성량은 30%이다. 메모리는 실험 장비의 실제 메모리가 2GB이므로 1.5배 크기인 3GB가 사용되었다. 메모리 위주 부하 생성을 위한 프로세스는 운영체제로 하여금 실제 메모리와 스왑 메모리 사이에 데이터 교환 작업을 수행하도록 하기 때문에 반복 실험을 수행해본 결과 모든 실험에서 생성된 부하 크기의 ±5% 범위 값까지 유지할 수 있었다.

(그림 5)(a)에서 사용자가 입력한 수치에 도달한 시점부터 실험 종료 시점까지의 부하 크기에 대한 산술 평균값이 34.77%로 나타나 사용자가 입력한 수치인 30%와는 4.77%의 오차가 발생하였으며, 이는 오차 범위 ±5%를 만족하여 적절한 부하 생성이 되었음을 의미한다. (그림 5)(a)에서 부하를 만듦과 동시에 90% 이상의 "user" 항목의 값을 보이는데, 이는 메모리에 적재될 임의의 데이터를 생성하기 때문이다.



(a)



(b)

(그림 5) 메모리 위주 부하 생성 방식의 실험 결과

(그림 5)(b)는 실제 메모리와 스왑 메모리 사이의 데이터 교환 크기를 시간의 변화에 따라 나타낸 것이다. "swap read"의 값은 10여분 이후부터 "swap write"의 값과 유사한 값이 발생되어 (그림 5)(b)에서 두 그래프가 중복되게 보인다. 0~5분 사이는 실제 메모리에 데이터를 적재하는 과정이므로 스왑 메모리와의 페이지 교환이 거의 발생되지 않는다. 실제 메모리를 모두 채운 5분 이후부터는 스왑 메모리로 많은 양의 데이터가 스왑 아웃된다. 메모리 위주 부하를 만들기 위한 준비 단계가 끝난 뒤인 10여분 이후부터는 소량의 데이터 교환만이 발생된다. 이를 통해, 운영체제가 메모리 관리를 위해 스왑 메모리와 실제 메모리 사이에서 데이터 교체 작업을 수행하고 있다는 것을 알 수 있다. 이는 메모리 위주 부하가 정상적으로 생성됨을 의미한다.

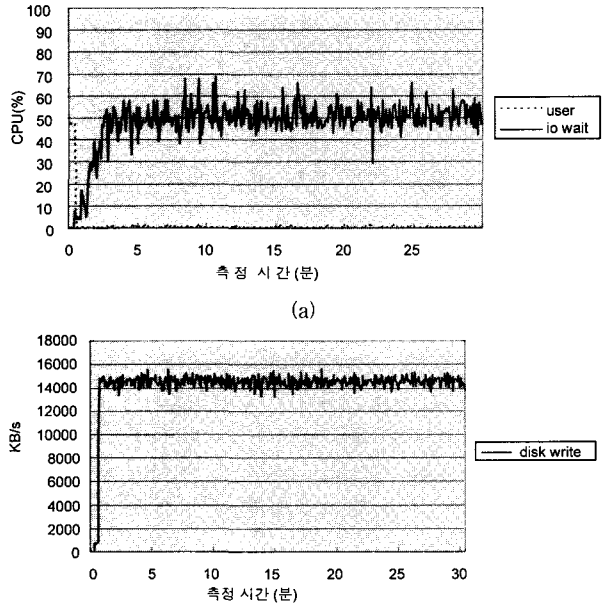
3.2 입출력 위주 부하 생성 실험

입출력 위주 부하 실험에 사용될 4개 파일의 각 크기는 5MB, 10MB, 15MB, 20MB로 설정하였다. 가장 작은 5MB 데이터 파일을 사용하여 입출력 부하를 만드는 경우 하나의 프로세스 당 평균 약 3%의 "io wait" 값이 발생되었고, 가장 큰 데이터 파일인 20MB 크기의 파일을 사용하는 경우에는 평균 약 7%의 "io wait" 값이 발생되었다. 5MB와 10MB 크기의 데이터 파일을 사용하는 경우에는 5% 이상의 CPU 작업 부하가 추가로 생성되어 휴지 시간을 최소 입력 단위인 1초로 사용하고, 15MB와 20MB 크기의 데이터 파일을 생성하

여 부하를 만드는 경우에는 1% 미만의 CPU 작업 부하가 추가로 생성되기 때문에 별도의 휴지 시간을 사용하지 않는다.

(그림 6)은 다양한 부하 생성을 위해 여러 가지 부하 수치를 입력하여 실험한 결과들 중 하나이며, 입력한 입출력 부하의 크기가 50%일 때 입출력 위주 부하 생성 실험의 결과를 보인다. 입출력 위주 부하의 크기는 반복 실험을 수행해 본 결과 모든 실험에서 생성된 부하 크기의 ±3% 범위 값까지 유지할 수 있었다.

(그림 6)(a) 그래프에서 "io wait" 값이 50%에 도달한 시점인 약 3분 이후부터 실험이 종료된 시점까지 만들어진 부하의 평균 크기는 약 51.01%이고, 이 값은 오차 범위를 만족한다. 입출력 위주 부하를 만들 때 여러 개의 프로세스들이 파일의 복사와 삭제 작업을 반복하게 됨에 따라, 복사 및 삭제 작업이 순차적으로 진행되지 않고 운영 체제의 상황에 따라 지연되거나 일괄적으로 처리될 수 있기 때문에 (그림 6)의 결과 그래프 (그림 6)(a)는 균일한 값을 보여주지 못한다.



(a)

(b)

(그림 6) 입출력 위주 부하 생성 방식의 실험 결과

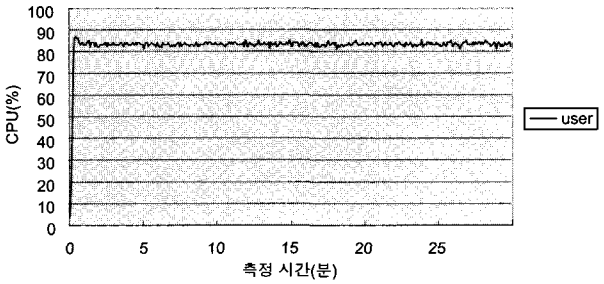
(그림 6)(b)는 시간의 경과에 따른 디스크 입출력의 크기를 나타내며, 초당 약 14MB 이상의 데이터가 디스크에 기록되었음을 보인다. 입출력 위주 부하 생성을 위한 데이터 파일은 처음 읽혀진 이후 캐시 메모리에 저장되기 때문에 "disk read"는 거의 발생하지 않아서 (그림 6)(b)에서 관련 부분을 생략하였다. (그림 6)의 그래프 (그림 6)(a)와 (그림 6)(b)를 통해 입출력 부하가 정상적으로 만들어져 데이터가 디스크로부터 입출력되고 있음을 알 수 있고, 입출력 데이터를 기다리기 위해 소비되는 CPU 사용량인 "io wait" 값을 통해 CPU가 디스크의 입출력을 위한 대기 시간으로 일정 시간을 소비하고 있음을 알 수 있다.

3.3 CPU 위주 부하 생성 실험

CPU 위주 부하의 크기 측정 시 "user" 항목 값과 "sys" 항목 값의 3초당 측정치에 대해 산술 평균을 구하여 반영한다. CPU 위주 부하의 크기는 반복 실험을 수행해본 결과 모든 실험에서 생성된 부하 크기의 ±1% 범위 값까지 유지할 수 있었다.

단위 작업 중 휴지 시간은 1초로 설정하였으며, 보정계수 값은 2.45로 설정하였다. 실험 장비에서 CPU 개수는 2개, CPU 클락 주파수는 400MHz, 시스템 클락 주파수는 100MHz 이므로 앞서 설명한 식 (1)에 의해 하나의 프로세스는 196,000번의 단순 반복 연산을 수행한다.

(그림 7)은 부하 생성을 위한 입력 값으로 여러 가지 수치를 입력하여 수행한 실험 결과들 중 하나이며, 사용자가 입력한 CPU 위주 부하의 생성량은 83%이다. CPU 위주 부하 크기 표시를 위해 사용되는 "sys" 항목과 "user" 항목 중 "sys" 항목의 값은 거의 측정되지 않아 (그림 7)에서 생략하였다.



(그림 7) CPU 위주 부하 생성 방식의 실험 결과

생성된 평균 CPU 위주 부하 크기는 83.16%로 설정 오차 범위 ±1%를 만족하였다. (그림 7)을 통해 CPU 위주 부하를 사용자가 지정한 크기만큼 오차 범위 내에서 정상적으로 생성함을 알 수 있다. 또한, 모든 사용자 프로세스들의 CPU 사용량을 나타내는 "user" 항목 값을 통해 CPU가 단순 반복 연산을 수행하는데 일정 시간을 할애하고 있음을 알 수 있다.

4. 통합 모듈의 실험 결과 및 분석

본 장에서는 벤치마크 시험에의 적용 실험에 대한 결과 및 분석 내용을 기술한다. TPC-C 벤치마크와 Wisconsin 벤치마크를 사용하여 부하 생성 및 벤치마크 시험에의 적용을 실험한다. TPC-C 벤치마크의 부하를 실세계 환경이라 가정하고, Wisconsin 벤치마크는 사용자들이 측정할 벤치마크로 가정한다. 실험은 앞장에 기술된 것과 동일한 서버를 사용하여 동일한 환경에서 수행하였다. 데이터베이스 시스템으로는 TPC-C 벤치마크와 Wisconsin 벤치마크의 적절한 수행을 위해 오라클 사에서 개발된 Oracle 9i를 사용하였다.

실험은 크게 두 가지로 나뉜다. 하나는 실세계 부하와 유사한 부하를 부하 생성 도구를 통해 생성할 수 있음을 보이는 실험이다. TPC-C 벤치마크 수행 시 데이터베이스의 크기를 조절할 수 있는 확장 요소(scaling factor)에 따라 부하의 크기를 측정하고, 측정된 부하 크기를 기반으로 부하 생성 도구를 통

해 유사한 부하를 생성한다. 다른 하나는 실세계 작업 부하 상에서 벤치마크 시험을 수행한 경우와 작업 부하 생성 도구를 통해 실세계 작업 부하와 유사하게 생성한 부하 상에서 벤치마크 시험을 수행한 경우를 비교하여 벤치마크 시험 수행 시 부하 생성 도구의 적용 가능 여부를 알아보는 실험이다.

실세계 부하와 유사한 부하를 발생시키기 위해서는 각 부하 생성 방식에 필요한 실세계 부하의 통계 정보 값을 미리 알고 있어야한다. 메모리 위주 부하를 발생시키는 경우에는 스왑 메모리와 실제 메모리 간의 치환된 데이터 양을 알아야 한다. 메모리 위주 부하 값은 (그림 5) (b)에서 치환된 데이터 약 16KB 당 1%의 "io wait" 값이 발생되었으므로 치환된 데이터 양에 따라 "io wait" 값으로 변환하여 입력한다. 입출력 위주 부하를 발생시키는 경우에는 스왑 메모리와 실제 메모리 간의 치환된 데이터 양과 "io wait" 값을 알아야 한다. 이때, "io wait" 값에는 메모리 위주 부하의 값이 포함되어 있으므로 입출력 위주 부하 값은 측정된 "io wait" 값에서 메모리 위주 부하 값을 제외하고 입력한다. CPU 위주 부하를 발생시키기 위해서는 "top" 명령어의 수행을 통해 알 수 있는 "user" 항목 값과 "sys" 항목 값을 알아야 하며, 부하 값 입력 시 "user" 항목 값과 "sys" 항목 값의 합을 입력한다.

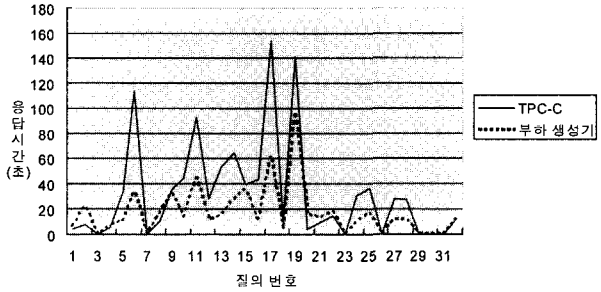
〈표 1〉 TPC-C 벤치마크와 부하 생성 도구를 각각 수행하여 생성된 부하 값

통계 정보 / 확장 요소	user+sys		오차	io wait		오차
	TPC C	부하 생성 도구		TPC C	부하 생성 도구	
Warehouse 5개	10.6%	10.4%	0.2%	11.3%	13.0%	1.7%
Warehouse 10개	25.1%	25.1%	0.0%	27.0%	24.9%	2.1%
Warehouse 15개	30.9%	29.0%	1.9%	40.4%	44.2%	3.8%

<표 1>은 TPC C 벤치마크와 부하 생성 도구를 각각 수행하여 생성된 부하 값을 나타낸다. <표 1>에서 "TPC-C" 열은 TPC-C 벤치마크 수행 시 확장 요소인 warehouse 개수를 5개, 10개, 15개로 설정하였을 경우의 통계 정보 값을 나타낸다. Warehouse 15개는 실험 대상 서버에서 수행할 수 있는 최대 확장 요소 값이다. "swap read"와 "swap write"의 값은 거의 측정되지 않았기 때문에 <표 1>에서는 생략하였으며, 메모리 위주 부하는 거의 발생하지 않았다는 것을 알 수 있다. <표 1>에서 warehouse 개수가 늘어날수록 부하 값은 전체적으로 증가하였으며, 이중 "io wait" 값이 가장 크게 증가하였다.

<표 1>에서 "부하 생성 도구" 열은 "TPC-C" 열의 값을 부하 표시기에 설정한 후 부하 생성 도구를 통해 부하를 생성하여 얻는 통계 정보 값을 나타낸다. 메모리 위주 부하는 거의 발생하지 않았으므로 부하 표시기에 입출력 위주 부하 값과 CPU 위주 부하 값만 입력하였다. 입출력 위주 부하 값으로는 "io wait" 값을 입력하였으며, CPU 위주 부하 값으로는 "user" 항목과 "sys" 항목 값의 합을 입력하였다. <표 1>에서 "오차" 열은 "TPC-C" 열값과 "부하 생성 도구" 열값 사이의 오차를 나타낸다. 부하 생성 도구를 통해 생성된 부하 값은 TPC-C 벤치마크 수행 시 생성된 부하와 유사한 결과를 보임을 알 수 있다.

(그림 8)은 TPC-C 벤치마크와 부하 생성 도구를 각각 Wisconsin 벤치마크와 연동 수행한 결과를 나타내는 그래프이다. TPC-C 벤치마크의 경우에는 벤치마크 수행 시 발생된 부하 상에서 Wisconsin 벤치마크를 연동 수행하여 각 질의 별 응답 시간을 측정하였으며, 부하 생성 도구의 경우에는 <표 1>에 기술된 부하 표시기 값을 입력하여 부하를 생성한 상태에서 Wisconsin 벤치마크와 연동 수행하여 각 질의 별 응답 시간을 측정하였다. 각 질의 별 응답 시간은 같은 실험을 10번 수행하여 얻은 응답 시간의 산술 평균 값으로 표시하였다.



(그림 8) TPC-C 벤치마크와 부하 생성 도구를 각각 Wisconsin 벤치마크와 연동 수행한 결과

(그림 8)에서 Wisconsin 벤치마크를 TPC-C 벤치마크와 연동하였을 경우의 결과 그래프와 부하 생성 도구와 연동하였을 경우의 결과 그래프는 전체적으로 유사한 형태를 보이지만 부분적으로는 약간의 차이가 있다. 이는 데이터베이스 시스템의 사용 정도에 따른 차이이다. TPC-C 벤치마크와 부하 생성 도구는 비슷한 값의 부하를 생성하지만, TPC-C 벤치마크는 데이터베이스 시스템에 대해 작업을 수행하고 부하 생성 도구는 그렇지 않다. 그러나 Wisconsin 벤치마크는 TPC-C 벤치마크와 같은 데이터베이스 시스템을 사용하여 수행되기 때문에 부하 생성 도구와 TPC-C의 그래프 차이가 생긴다.

5. 결 론

본 논문에서는 데이터베이스 시스템 벤치마크가 일반적으로 비현실적인 실험 환경에서 수행됨을 지적하고 현실적인 환경에서 수행되도록 실세계 부하와 흡사한 부하를 제공하는 부하 생성 도구를 기술하였다. 데이터베이스 시스템 벤치마크를 수행함에 있어 실세계 부하에 가까운 실험 환경은 벤치마크의 신뢰도를 높이는데 중요한 요소가 된다. 이를 위해 실세계 업무 시스템에서 발생되는 부하의 종류를 메모리 위주 부하, 입출력 위부 부하, CPU 위주 부하로 구분하였으며, 각 부하의 특성을 고려한 생성 방법을 보였다. 부하 생성 도구는 각 부하의 크기를 사용자가 조정하여 부하를 생성함으

로써 다양한 부하를 생성할 수 있다.

부하 생성 도구를 통한 부하 생성의 적절성을 검증하기 위해 각 부하 모듈별 부하 생성 실험을 수행하였다. 또한, 부하 생성 도구와 Wisconsin 벤치마크와의 연동 실험과 실세계 부하 상에서의 연동 실험을 수행하여 부하 생성 도구가 데이터베이스 시스템 벤치마크를 위한 부하 생성 도구로서 그 역할을 수행할 수 있음을 보였다. 부하 생성의 적절성 여부를 위한 실험은 하드웨어 구성이 다른 2개의 서버에서 수행을 하였으며, 그 결과 사용자가 지정한 부하 크기 값에 따라 적절한 부하를 생성하였기 때문에 서버 종류에 무관하게 부하 생성 도구를 활용할 수 있음을 알 수 있다.

본 부하 생성 도구는 데이터베이스 시스템 벤치마크뿐만 아니라 일반 응용 프로그램의 벤치마크 및 부하 생성이 필요한 다양한 분야에서 활용될 수 있을 것이다. 일반 응용 프로그램의 벤치마크 및 다양한 분야에서의 활용에 대한 적절성을 보이기 위해 많은 응용 프로그램을 대상으로 추후 실험을 수행하고, 그 실험 결과를 분석하여 해당 분야에서 필요한 부하 생성이 적절히 수행됨을 보일 것이다. 현재 특정 형태의 부하를 유사하게 생성하기 위해서는 부하 생성과는 별도로 통계 정보 수집을 위한 작업이 추가로 필요하며, 운영 체제가 제공하는 명령어에 의존하여 수작업으로 이루어진다. 통계 정보 수집 및 부하 생성 작업의 자동화를 통해 사용자가 특정 형태의 부하 생성을 손쉽게 수행할 수 있도록 관련 모듈의 개발이 향후 필요하다. 통계 정보의 실시간 관리를 통해 생성된 부하의 변화를 분석하면 추후 유사한 부하 생성 시 큰 도움이 될 수 있다.

참 고 문 헌

[1] A. B. Chaudhri, "An Annotated Bibliography of Benchmarks for Object Databases," ACM SIGMOD Record, 24(1), pp.50-57, 1995.

[2] P. Martin, W. Powely, H. Y. Li and K. Romanufa, "Managing Database Server Performance to Meet QoS Requirements in Electronic Commerce Systems," International Journal on Digital Libraries, 3(4), pp.316-324, 2002.

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

[4] S. H. Lee, S. J. Kim and W. Kim, "The BORD Benchmark for Object-Relational Databases," Proceedings of the 11th Database and Expert Systems Applications Conference, pp.6-20, 2000.

[5] P. O'Neil, "The Set Query Benchmark," In : The Benchmark Handbook : pp.359-396, J. Gray Ed., Morgan Kaufmann, 1993.

[6] D. DeWitt, "The Wisconsin Benchmark : Past, Present, and Future," In : The Benchmark Handbook : J. Gray Ed., Morgan Kaufmann, pp.269-316, 1993.

[7] D. G. Feitelson, "Workload Modeling for Performance Evaluation," International Symposium on Computer Modeling, Measurement and Evaluation, pp.114-141, 2002.

[8] M. Calzarossa and G. Serazzi "Workload Characterization : A Survey," Proceedings of the IEEE, 81(8), pp.1136-1150, 1993.

[9] Cray Inc., "Optimizing Code on Cray PVP Systems," Cray Research Online Software Publication : SG-2192, 1997.

[10] W. W. Hsu, A. J. Smith and H. C. Young, "Analysis of the Characteristics of Production Database Workloads and Comparison with the TPC Benchmarks," Technical Report, IBM Almaden Research Center, 1999.

[11] W. W. Hsu, A. J. Smith and H. C. Young, "Characteristics of Production Database Workloads and the TPC Benchmarks," IBM Systems Journal, 40(3), pp.781-802, 2001.

김 기 욱

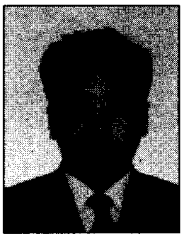
e-mail : k2www@dreamwiz.com

2002년 강남대학교 산업, 전자, 전산 공학부 학사

2004년 숭실대학교 대학원 컴퓨터학과 석사

2004년~현재 (주)아이티네이드 전자인증 사업팀

관심분야 : 부하 생성기, 부하, 벤치마크



정 회 진

e-mail : sinclear@dreamwiz.com

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

관심분야 : 데이터베이스 시스템 성능 평가 및 튜닝, XML 데이터베이스

이 상 호

e-mail : shlee@comp.ssu.ac.kr

1984년 서울대학교 전산공학과 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

관심분야 : 인터넷 데이타베이스, 데이타베이스 시스템 성능 평가 및 튜닝