데이터베이스 시스템 벤치마크를 위한 부하 생성기 설계

김기욱*, 정회진*, 이상호**
* 숭실대학교 대학원 컴퓨터학과
** 숭실대학교 컴퓨터학부
e-mail: k2www@dreamwiz.com

Design of a Workload Generator for Database System Benchmarks

Kee Wuk Kim* Hoe Jin Jeong* Sang Ho Lee**
* Department of Computing, Soongsil Graduate School
** School of Computing, Soongsil University

요 약

현대 사회에서 사용되는 많은 데이터베이스 시스템 벤치마크에서는 결과 값의 극대화를 위해 실험 대상 시스템의 가용 자원을 최대화한 상태에서 수행하는 문제점을 가지고 있다. 실세계에서의 작업 환경과 유사한 환경에서의 벤치마크 실험을 위해 본 논문에서는 기존 벤치마크를 보완할 수 있는 부하 생성기를 설계한다. 부하 생성기는 운영체제의 메모리와 디스크, CPU에 직접적인 부하를 생성하며, 실세계 부하와 유사하고 사용자들이 쉽게 조작 가능한 통합 부하 생성을 지원한다.

1. 서 론

현대 사회에 응용되고 있는 모든 시스템 및 소프트웨어는 개발 단계에서부터 사용자의 선택에 이르기까지 많은 실험을 통한 평가를 필요로 한다. 데이터베이스 시스템 또한 실세계의 업무 영역을 잘 지원하고 있는지 평가할 필요가 있으며, 데이터베이스 시스템 벤치마크가 이를 위해 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, 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, 웹 전자 상거래 분야에는 TPC-W 벤치마크[3]가 있다.

데이터베이스 시스템 벤치마크는 신뢰할 수 있는 실험 결과를 위해 실세계의 해당 분야를 대표할 수 있는 부하를 사용하여야 한다. 그러나 기존 데이터베이스 시스템 벤치마크들은 결과 값의 극대화를 위해 실험 대상 시스템의 불필요한 응용 프로그램을 종료하고, 가용 자원을 최대화한 상태에서 수행한다는 한계성을 지니고 있다. 이런 한계성을 극복하기 위해서는 실세계에서의 작업 환경과 유사하게 만들어 기존의 성능 평가와는 다른 환경에서의 성능 평가를 수행토록 하는 것이 필요하다. 이를 위해서 기존 데이터베이스 시스템 벤치마크를 보완할 수 있는 부하 생성기가 필요하다. 본 논문은 데이터베이스 시스템 벤치마크를 위한 부하 생성기의 설계 내용을 기술한다.

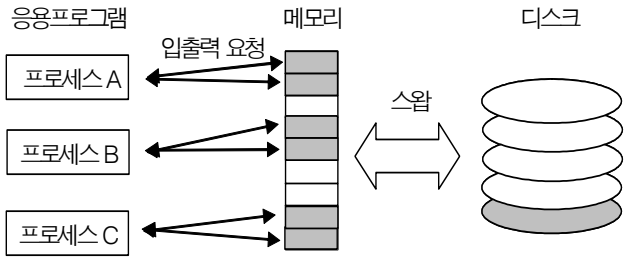
2. 부하의 분류 및 생성 방법

실세계 업무 시스템에서 발생되는 다양한 부하에 대해 그 종류를 분류하면, 메모리 위주 부하, 입출력 위주 부하, CPU 위주 부하로 나뉘어 질 수 있다. 부하의 분류 및 특성 확인을 위해 운영 체제의 “top” 명령어와 “iostat” 명령어에서 제공하는 기본 통계

정보 값을 사용한다. “top” 명령어는 사용자 당 프로세스 단위의 자원 사용량과 시스템 전체의 자원 사용량을 사용자가 쉽게 알 수 있도록 여러 가지 통계 항목들을 제공한다. 여러 통계항목 중에서 “user” 항목 값은 모든 사용자 프로세스들의 CPU 사용량을 나타내고, “sys” 항목 값은 커널(kernel)의 CPU 사용량을 나타내며, “io wait” 항목 값은 입출력 데이터를 기다리기 위해 소비되는 CPU 사용량을 나타낸다. CPU 사용량은 각 항목 별 측정 대상 프로세스들의 CPU 사용 시간을 비교 수치인 “%”로 나타낸 값이다.

2.1 메모리 위주 부하

메모리 위주 부하는 사용자 프로세스가 실제 메모리 공간 중에서 가용 메모리 공간보다 많은 양의 메모리를 사용함으로 인해 운영 체제가 치환(swapping) 작업을 통해 메모리를 관리하는 일을 말한다. 스왑 메모리와 실제 메모리는 상호간에 치환을 쉽게 하기 위해 페이지라는 같은 크기의 작은 조각으로 나뉜다[7]. 운영 체제는 프로세스가 스왑 메모리에 있는 페이지를 실제 메모리로 가져올 때, 실제 메모리에 빈 공간이 없다면 빈 공간을 만들기 위해 LRU(least recently used) 알고리즘에 따라 실제 메모리에서 제거될 페이지를 찾는다. 메모리 위주 부하는 <그림 1>과 같이 두 개 이상의 프로세스가 실제 메모리 공간을 대상으로 경합할 때 운영 체제가 실제 메모리 상에 빈 공간을 만드는 과정에서 생성된다.



<그림 1> 메모리 위주 부하

본 논문에서는 메모리 위주 부하의 생성을 위해 데이터 생성 작업과 생성되는 데이터를 메모리에 적재하는 작업부터 수행하며, 두 작업은 동시에 이루어진다. 이때, 실제 메모리가 다 채워진 이후에는 나머지 데이터를 메모리에 적재하기 위해 운영 체제가 실제 메모리에 있는 일부 데이터를 스왑 메모리와 치환한다. 데이터 적재 과정이 완료되면 스왑 메모리에 적재된 데이터를 실제 메모리로 불러오기 위해 단순 연산을 수행한다.

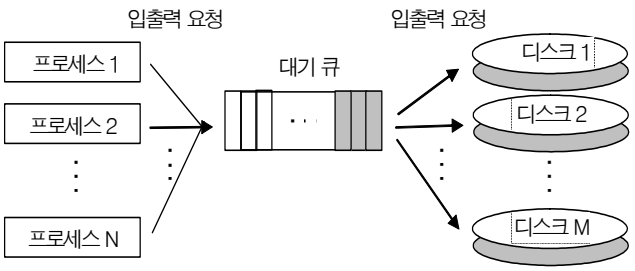
메모리 위주 부하의 생성량은 사용자가 입력한 값에 기반을 둔다. 하지만, 사용자가 운영 체제가 제어하는 자원을 사용하여 정확하게 제어하는 것은 어렵기 때문에 사용자 입력 값만큼의 메모리 위주 부하를 특정 시각에 생성할 수는 없다. 따라서 부하의 생성량을 계속 조정하며 유지하여야만 한다. 이를 위해서 메모리 위주 부하 생성 프로세스는 스왑 메모리로부터 일정 크기의 데이터를 읽어올 때마다 휴지 시간(sleep time)을 갖는다. 각각의 프로세스들이 짧은 휴지 시간을 가지면서 스왑 메모리 상의 데이터를 읽어 들이면 생성하는 부하의 양은 증가하고, 긴 휴지 시간을 갖는 경우에는 부하의 생성량은 감소한다.

생성되는 메모리 위주 부하의 양은 운영체제에서 제공하는 통계 정보 값 중 ″io wait″ 항목 값과 “iostat” 명령어의 결과 값을 통해 알 수 있는 실제 메모리와 스왑 메모리 사이의 데이터 교환 크기를 통해서 알 수 있다. 실제 메모리와 스왑 메모리 사이의 데이터 교환 크기는 실제 메모리로의 스왑 인(swap in)과 스왑 아웃(swap out)된 데이터의 크기를 나타내며, 스왑 인된 데이터의 크기와 스왑 아웃된 데이터의 크기는 비슷하게 나타난다. 메모리 위주 부하의 양은 사용자가 쉽게 이해할 수 있도록 “%” 단위를 사용하는 “io wait” 항목 값으로 나타내며, 생성되는 메모리 위주 부하의 양은 사용자가 입력한 메모리 위주 부하의 양과 ±5%의 오차 범위를 갖는다. 메모리 위주 부하의 양을 측정하기 위해 “io wait” 항목 값에 대해 3초당 측정치의 산술 평균값을 사용하였다. 이는 메모리 위주 부하를 생성할 때 활발한 치환이 발생되면서 “io wait”의 값이 일정하지 않고 약간의 변동이 생기기 때문이다.

2.2 입출력 위주 부하

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. 입출력 요청이 디스크로 전달될 때, 하나의 요청씩 순차적으로 전달되지 않고 여러 개의 요청이 동시에 디스크에 전달될 수도 있다. 입출력 위주 부하는 <그림 2>에서 보이듯이 하나 이상의 프로세스가 하나 이

상의 디스크에 대해 입출력 작업을 요청할 때 생성된다.



<그림 2> 입출력 위주 부하

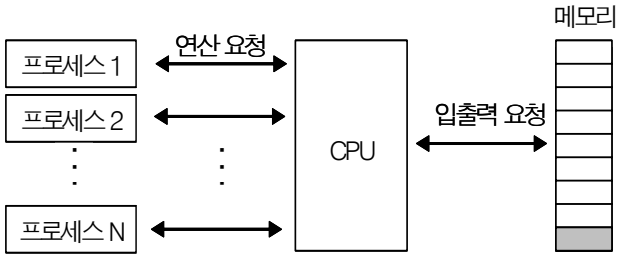
입출력 위주 부하는 디스크에서 일정 크기의 데이터 파일들을 복사하고 삭제하는 작업을 통해 생성되며, 그 양은 데이터 파일의 복사와 삭제 작업을 수행하는 프로세스의 수와 데이터 파일의 크기를 변경하여 조절한다. 입출력 위주 부하의 양은 통계 정보의 “io wait” 항목 값을 사용한다.

입출력 위주 부하 생성은 작업 프로세스 1개를 생성하여 해당 프로세스가 데이터 파일을 사용자로부터 입력받은 디렉터리 경로 상에 복사 및 삭제 작업을 수행함으로써 이루어진다. 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 큰 경우에는 사용하였던 데이터 파일보다 작은 크기의 데이터 파일을 사용하여 복사 및 삭제 작업을 새로 수행하여 부하의 크기를 측정한다.

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

2.3 CPU 위주 부하

CPU 위주 부하는 사용자 프로세스가 CPU로 하여금 많은 계산을 수행하게 하는 일을 말한다. CPU 위주 부하는 여러 개의 프로세스가 CPU로 하여금 계산을 수행하도록 연산 요청을 할 때 만들어진다. <그림 3>은 CPU 위주 부하를 만드는 방법을 나타낸다.



<그림 3> CPU 위주 부하

CPU 위주 부하를 만들기 위해 하나의 프로세스는 단순 증가 연산 작업과 휴지 시간 1초를 단위 작업으로 하여 작업을 수행한다. 단순 증가 연산 작업은 (식 1)에서 계산된 횟수만큼 반복 수행된다. CPU 위주 부하의 측정은 통계 정보의 “user” 항목 값을 사용한다.

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

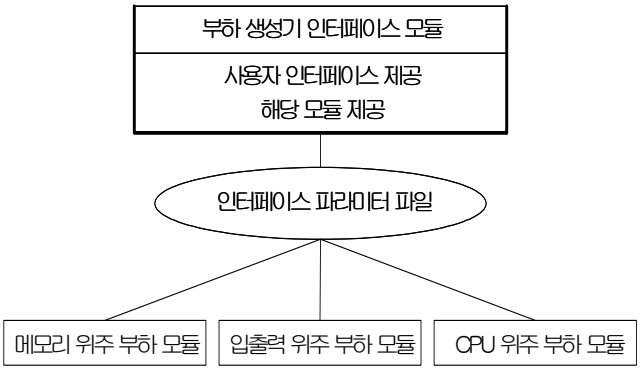
운영체제는 사용자가 입력한 CPU 위주 부하 양에 따라 CPU 위주 부하 생성을 위해 작업을 수행할 프로세스를 N개까지 생성한다. 작업 프로세스의 수를 1개부터 N개까지 증가시키며 부하 생성 작업을 수행하여 그 측정치를 구하는 경우에는 부하 생성 완료까지 많은 시간이 걸릴 수 있다. 이를 보완하기 위해 일정 부하 크기까지는 프로세스를 일괄 생성하여 작업을 수행한 후 측정치를 구하고, 구한 측정치가 사용자가 입력한 부하의 양보다 크거나 작은 경우에는 프로세스를 한 개씩 증가시키거나 감소시키면서 보정하는 방법을 사용한다.

CPU 위주 부하의 양은 통계정보에서 사용자 CPU 사용량을 나타내는 “user” 항목 값과 운영체제 커널의 CPU 사용량을 나타내는 “sys” 항목 값의 합으로 측정하고, 3초당 측정치에 대해 산술 평균을 구하여 반영한다. CPU 위주 부하를 생성한 후 측정된 값은 사용자가 입력한 CPU 위주 부하의 목표 값과 약 ±1%의 오차 범위 내에서 그 값을 가진다.

3. 부하 생성기 모듈 구성

부하 생성기는 다중 프로세스 기반 운영 체제에서 수행되며, 모듈 구성은 <그림 4>와 같다. 메모리 위주 부하 모듈, 입출력 위주 부하 모듈, CPU 위주 부하 모듈은 각 부하들을 생성하는데 이용된다. 인

터페이스 파라미터 파일에는 부하 생성에 필요한 내용을 사용자로부터 입력 받은 내용이 기록된다. 부하 생성기 인터페이스 모듈은 각 모듈들을 통합하여 관리하고, 사용자 인터페이스를 담당한다.



<그림 4> 부하 생성기 모듈의 구성

인터페이스 파라미터 파일에는 모듈들이 위치한 기본 디렉터리 경로, 입출력 위주 부하 생성 시 사용할 디스크 개수와 이에 따른 디스크 경로, 부하 생성 시간이 저장된다. 각 경로 명은 절대 경로 명을 사용한다. 부하 생성 시간은 부하 생성기가 수행될 총 시간을 말하며 분 단위로 입력한다. 부하 생성기의 세 가지 모듈은 개별적으로 실행될 수도 있고 서로 조합하여 실행될 수도 있다. 부하 크기의 입력은 인터페이스 파라미터 파일에서 부하 표시기(workload indicator)를 통해 입력한다. 부하 표시기란 인터페이스 파라미터 파일 안에 있는 부하 입력 엔트리이다. 부하 표시기에는 메모리 위주 부하 모듈, 입출력 위주 부하 모듈, CPU 위주 부하 모듈을 통해 생성할 부하 크기를 순서대로 입력한다.

부하 생성기 인터페이스 모듈은 각 부하 생성 모듈에 인자 값으로 부하 생성 시작 시간과 인터페이스 파라미터 파일을 넘겨준다. 인자 값을 넘겨받은 각 모듈들은 인터페이스 파라미터 파일의 내용 중 필요한 부분만을 읽어와 부하 생성 작업을 수행하며, 일정 시간마다 수행된 시간을 계산하여 지정 시간 동안만 부하 생성 작업을 수행하도록 한다.

4. 결론 및 향후계획

본 논문은 실세계 업무 시스템에서 발생되는 부하의 종류를 분류하고, 각 분류 방식에 맞추어 부하를 생성할 수 있는 부하 생성기의 설계 내용을 기술하였다. 부하 생성기의 부하 생성 방식을 메모리 위주 부하, 입출력 위주 부하, CPU 위주 부하로 구분하였으며, 세 가지 종류의 부하 크기를 각각 조절하여 부하를 생성함으로써 실세계 업무 시스템의 부하와 유사한 부하를 생성할 수 있도록 설계하였다. 데이터베이스 시스템 벤치마크를 수행함에 있어 실세계 부하에 가까운 실험 환경은 벤치마크의 신뢰도를 높이는데 중요한 요소가 된다.

앞으로 본문의 설계 내용에 대해 타당성을 알아보는 실험을 수행한 후 각 모듈을 구현 할 계획이다. 구현이 완료된 후에는 특정 데이터베이스 시스템 상에서 OLTP 분야 벤치마크인 TPC-C를 부하 생성기와 연동하여 부하 생성기의 성능에 대한 실험을 수행할 계획이다. 추후에는 여러 데이터베이스 시스템 상에서 OLTP 분야뿐만 아니라 DSS 분야의 벤치마크와 웹 전자 상거래 분야의 벤치마크를 부하 생성기와 연동하여 부하 생성기가 실세계 부하를 얼마만큼 잘 반영하는지 알아보는 연구를 진행할 것이다.

참고 문헌

[1] A.B. Chaudhri, “An Annotated Bibliography of Benchmarks for Object Databases”, ACM SIGMOD Record, Vol. 24, No. 1, pages 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, Vol. 3, No. 4, pages 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, 11th Database and Expert Systems Applications Conference, pages 6-20, 2000.

[5] P. O’Neil, “The Set Query Benchmark”, The Benchmark Handbook, pages 359-396, J. Gray Ed., Morgan Kaufmann, 1993.

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

[7] D.A. Rusling, “The Linux Kernel”, The Linux Documentation Project, 1996.