임의 추출 분할 방식을 이용한 동적 문제 출제 시스템

원대희° 강태호 김원진 방 훈 이재영
한 림 대 학 교 컴 퓨 터 공 학 과
{dhwon, Lamius, ,wjkim hooni, jylee}@isul.ce.hallym.ac.kr

A System to Generate Dynamic Test Using a Random Sampling Division Method

D.H Won° T.H kang H.Bang W.I kim J.Y. Lee
Dept. of Computer Engineering, Hallym University

요 약

원격 교육에서 시간과 장소에 제한을 받지 않도록 하기 위해서 비슷한 난이도의 문제를 출제할 수 있는 동적 문제 출제 시스템이 필요하다. 이 시스템은 웹 서버에서 문제를 임의 추출하고, 추출된 문제를 임의 정렬하여, 정렬된 문제를 출력하는 방법을 사용하였다. 이 방법은 서버의 부하를 가중시켜 실행시간을 증가시키는 문제점이 있다. 이러한 문제점을 해결하기 위해서 임의 추출 기능들을 데이터베이스 서버, 웹 서버, 클라이언트로 분할하여 실행시간을 단축시키는 시스템을 제안하고 구현하였다.

1. 서 론

원격교육에서 학습자는 자신이 편한 시간에 편리한 장소에서 학습을 하고 평가를 받는다. 이런 교육에서 학습자의 학습 능력을 평가할 때, 같은 문제로 여러 학습자를 평가하면 문제가 유출되어 공정하고 정확하게 능력을 평가하기 어렵다. 이러한 문제점을 해결하기 위해서 비슷한 난이도의 문제를 다양하게 출제하는 동적 문제 출제 시스템이 제안되었다[1, 2, 3].

이 시스템은 임의 추출 방식을 채택하여 문제의 출력과 순서 등을 매번 변화시킴으로써 정확한 성적의 평가를 이룰 수 있고, 또한 반복 학습으로 오는 지루함을 줄여 효율적으로 학습할 수 있게 된다[2]. 하지만 문제의 동적 출력을 위한 임의 추출은 웹 서버에 부하를 가중시키면서 이로 인하여 웹 페이지의 로딩 시간의 증가라는 단점이 발견되었다.

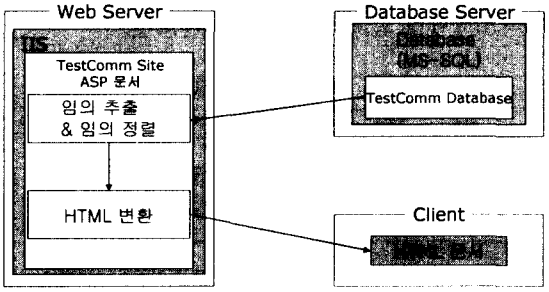
본 논문에서는 웹 프로그래밍 자체 내에서 임의 추출 방식을 이용한 종래의 출제 방법을 탈피하여 웹 서버, 데이터베이스 서버, 및 Client에 임의 추출 방식을 분산함으로써, 서버의 부담을 줄이고 웹 페이지의 로딩 시간을 최소화하여 로딩 시간을 개선한다.

2. 동적 문제 출제 시스템(SGDT)

2.1 구성도

동적 문제 출제 시스템(SGDT:System to Generate Dynamic Test)은 문제가 저장된 데이터베이스에서 문제를 임의 추출하고, 이 문제를 임의 정렬하여 HTML로 변환하는 기능을 웹 서버 내의 ASP문서가 모두 담당하였다. 객관식 문제의 경우 각 문제의 선택항들도 임의 정렬 정렬한다.

[그림 1]은 동적 문제 출제 시스템의 구성도로서, 데이터베이스 서버는 단순히 문제, 출제정보, 개인정보 등의 자료를 저장하는 역할만 하고, 클라이언트에서도 웹 서버로부터 온 문제를 화면에 출력한 후 답을 입력받아서 웹 서버로 되돌려보내는 간단한 기능을 수행한다.



[그림 1] SGDT의 시스템 구성도

단지 웹 서버에서만 개인 능력 정보를 분석하여 데이터베이스에서 문제를 임의 추출하고 그 문제를 임의로 정렬하고 필요하면 각 문제의 선택 항들도 임의로 정렬하는 등의 모든 기능을 수행 하고 있다.

2.2 문제 추출 알고리즘

SGDT는 각 학습자 능력 분석 프로그램을 이용하여 문제를 추출하고, 이 문제들은 다시 임의 추출과 임의 정렬하여 HTML문서의 형태로 클라이언트로 전송하였다.

학습자 분석 방식은 학습자의 이전 성적의 로그 자료를 바탕으로 각 평균 점수를 비교하여 낮은 부분부터 높은 부분으로 가중치를 백분율로 부여한 출제기준표를 작성한 후, 이를 이용하여 각 비율에 맞게 문제들을 임의 추출한다.

임의 추출 방법은 ASP 내부 함수인 Randomize를 사용하여 작성된 랜덤 표를 배열 변수에 저장하여 재사용하는 방법을 사용한다[2].

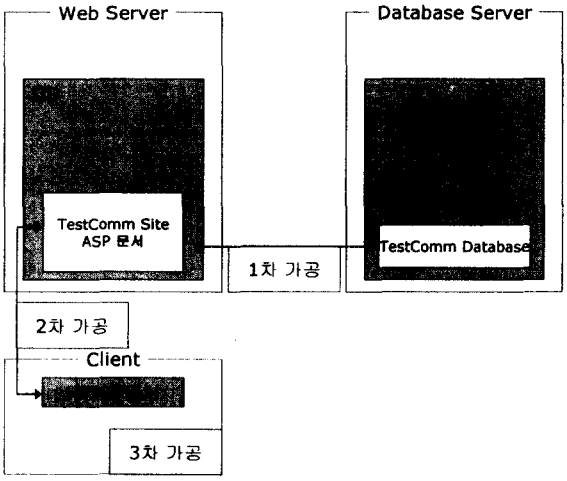
SGDT에서 사용한 임의 추출 방법은 랜덤의 횟수가 어느 정도 줄지만 문제의 개수에 따라 자칫 무한 루프가 될 수 있는 위험이 있고, 동시 접속자 수가 일정 인원이 넘으면 랜덤 함수에 의한 시스템의 과부하로 인해 웹 페이지의 로딩 속도가 현저히 느려지는 문제를 가지고 있다.

3. 임의 추출 분할 방식을 이용한 동적 문제 출제 시스템 (SRSDM)

SGDT에서 사용한 기존의 개인 능력 분석 방식은 그대로 따르면서 이전의 임의 추출 방식은 각 서버와 클라이언트에 각각의 기능을 분할하여 문제를 출제한다. 즉 문제 및 예제 문항의 임의 추출기능과 문제의 임의 정렬, 예제 문항의 임의 정렬의 기능을 분할하여 문제를 출력하여 준다.

3.1 동적 문제 출제 시스템의 구성

본 시스템은 SGDT의 기본 구성을 그대로 따르면서 웹 서버에 치중된 임의 추출 방식을 데이터베이스 서버와 클라이언트에 그 기능을 분배하였다. [그림2]는 개략적인 시스템 구성도를 보여준다.



[그림 2] SRSDM의 구성도

클라이언트에서 문제를 요구하면 웹 서버는 요구한 과목과 주차(주별)에 맞는 문제를 출력하기 위하여 데이터베이스 서버에 자료를 요청한다. 데이터베이스 서버는 이 자료를 보낼 때 문제 개수에 맞추어 임의 추출하여 1차 가공된 자료를 웹 서버로 되돌려 준다. 웹 서버에서는 1차 가공된 자료를 사용하여 ASP문서에 따라 자료를 임의 출력하여 동적으로 문제를 바꾸어 2차 가공된 자료를 HTML문서로 변환하여 클라이언트에 보내주면 마지막으로 HTML 문서 내에 포함된 자바 스크립트를 이용하여 예제 문항을 재정렬하여 마지막 3차 가공 후 완전한 문제 리스트가 출력된다.

3.2 임의 추출 분할을 사용한 문제 추출 알고리즘

클라이언트에서 문제의 출력 요청이 들어오면, 시스템은 요구에 맞는 문제를 다시 데이터베이스 서버에 요청을 하고 데이터베이스 서버는 요구에 맞게 임의 추출 후 가공하여 웹 서버로 보낸다. 다시 웹 서버에서는 ASP 문서의 서버 컴포넌트에 의한 임의 추출 방식을 사용하여 임의 정렬을 한 후에 클라이언트로 문제를 전송한다. 다시 클라이언트에서는 자바 스크립트에 의해 각 문제에 속한 예제 문항을 임의 정렬하여 최종적으로 브라우저를 통해 출력을 한다.

이때 서버와 클라이언트의 각각에서 임의 추출 알고리즘은 다음과 같다.

3.2.1 데이터베이스 서버에서의 임의 추출 알고리즘

현재의 SQL 문장에서는 랜덤함수가 존재하지 않으므로 임의 추출은 불가능하다. 하지만 MS-SQL에서 지원하는 Stored Procedures를 사용하면 임의 추출이 가능하다. Stored Procedures는 일종의 함수의 형태로 서버에 저장되어 미리 캐시에서 컴파일된 상태로 존재한다. 이것은 간단한 호출로 빠르게 실행되는데 일반 SQL문을 사용하는 것보다 월등히 빠르다[4]. Stored Procedures를 사용하여 문제의 개수만큼 재사용이 용이한 반면 이전 시스템에서는 매번 임의의 수를 문제의 개수만큼 추출해서 이를 웹 서버에 보내준다. 문제에 따른 각 예제 문항도 이와 같은 방법으로 웹 서버로 보내 준다. [소스1]은 Stored Procedures로 저장된 임의 추출 프로그램의 일부분이다.

```
SET @counter = (SELECT RAND(
    (DATEPART (mm, GETDATE()) * 100000)
  + (DATEPART(ss, GETDATE()) * 1000)
  + DATEPART(ms, GETDATE())))

SET @counter = @counter * 1000
SET @divider = @counter / 10
SET @temp = @counter
SET @counter = @temp % @divider
```

[소스 1] 데이터베이스 서버에서 구현한 임의 추출

일반적인 MS-SQL Server에서 제공하는 랜덤 함수를 이용하면 시스템에 의해 매번 같은 수가 도출되므로 DATEPART 함수를 이용해 임의의 소수가 출력된다. 다시 이 수를 이용하여 10보다 작은 수를 도출하여 완성된 임의의 수가 산출된다.

3.2.2 ASP문서에서의 임의 추출 알고리즘

데이터베이스 서버에서 임의 추출된 문제와 문항은 일정하게 정렬된 상태로 웹 서버에 전달된다. 따라서 이 자료를 임의적으로 재정렬해야 되는데, SGDT에서는 Randomize함수를 임의 추출 때마다 호출하여 사용한 반면 현재 시스템에서는 객체를 생성하여 이 객체를 이용해 임의 추출하였다. 이렇게 도출된 임의 수를 이용하여 문제만을 재정렬하고, 각각의 문제에 따른 예제 문항은 자바 스크립트의 배열 변수에 저장하게 한 후 클라이언트로 HTML 문서를 전송한다.

ASP문서에는 서버 컴포넌트가 존재하는데, 이것을 사용하면 일반 ASP 내에 직접 함수 호출하여 실행하는 시간보다 빠르게 실행되기 때문에 이 시스템에서는 서버 컴포넌트를 사용하였다[5].

```
Randomize
Random = int((var_number*Rnd)+1)
```

```
Set RandomSelect
        = Server.CreateObject("MSWC.Tools")
Abs( RandomSelect.Random ) Mod 10
```

[소스2]SGDT와 SRSDM의 임의 추출 부분 비교

[소스 2]는 기존의 랜덤 부분과 컴포넌트를 사용한 웹 페이

부분을 비교한 것이다. 아래의 Randomize 함수를 사용하여 임의의 수를 도출하는 부분보다 컴포넌트를 사용하는 상위의 코드가 더 복잡해 보이지만, 컴포넌트는 서버에 컴파일이 되어 DLL의 상태로 존재하기 때문에 실행이 빠르고 한번 객체로 생성하면 계속 재사용이 가능한 반면 이전 시스템에서 사용한 방식을 사용하면 매번 Randomize 함수를 호출하여 임의 수를 추출해야 되는 번거로움이 있다.

3.2.3 클라이언트에서의 임의 추출 알고리즘

웹 서버에서는 문제의 임의 정렬을 담당하는 반면, 클라이언트에서는 문제에 따른 예제 문항을 정렬하게 하여 임의 추출에 의한 부하를 분산하도록 했다. 서버에는 ASP문서가 자바 스크립트 문서를 생성하도록 하여 자바스크립트로 넘어온 예제 문항들을 각각의 배열에 저장 후 이를 임의 추출하여는 방식을 사용하였다.

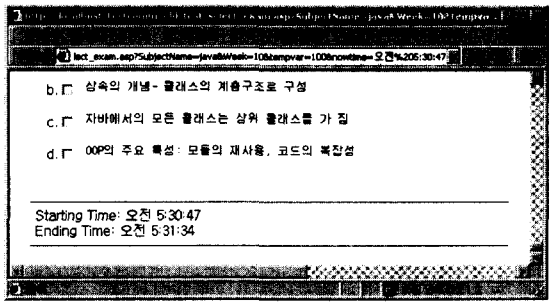
4 . 구현 및 비교 검토

4.1 시스템의 구현 환경

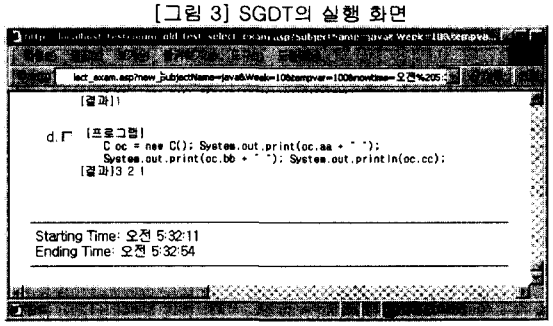
SRSDM의 시스템 구현하는 환경으로는 Windows 2000 Server와 MS-SQL Server 7.0을 사용하였으며, 웹 서버로는 IIS 5.0이 사용되었고, 프로그래밍 언어는 ASP와 자바 스크립트를 사용하여 작성되었다.

4.2 성능 분석 비교

SGDT와 SRSDM과의 실행 속도를 비교하기 위해 서버에서 ASP문서가 호출되어 client에 페이지가 로딩되기까지의 시간이 출력되는 프로그램을 작성하여 비교 분석하였다. 프로그램은 100번 동안 반복하여 브라우저에 출력하게 한 후 그 시간을 체크하도록 작성하여 비교하였다.



[그림 3] SGDT의 실행 화면

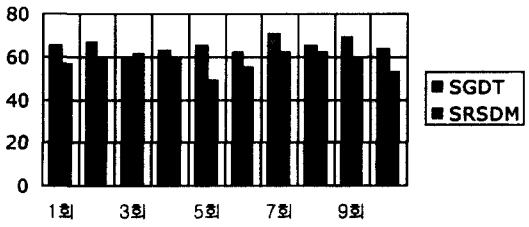


[그림 4] SRSDM의 실행 화면

지의 이미지와 HTML 태그는 모두 삭제하고 ASP 문서에서 바로 문제와 예제 문항이 출력이 되게 한 후 비교하였다. [그림 3]과 [그림 4]는 SGDT와 SRSDM을 각각 실행한 결과 출력 화면이다.

SGDT와 SRSDM을 비교하기 위한 실험 방법은 동일한 서버에 두 시스템을 번갈아 가면서 클라이언트에서 브라우저를 통해 실행하여 실행 시간을 체크하였고, 각각 총 10번을 번갈아 가며 실행하여 그 평균을 구하여 비교하였다. 좀더 정확한 비교를 위하여 매 번 실행 할 때마다 서버를 재부팅 한 후에 실험하여 결과를 도출하였다. [차트 1]은 SGDT와 SRSDM의 실행 시간을 보여주고 있다. SRSDM의 평균 실행 시간 약 이 57.7초로 이고 SGDT는 65.1초로 약 7.4초 가량 실행 시간이 빠르다는 결과가 나타났으며, 약12.3%의 속도가 향상이 되었다. 이 결과는 동시접속자수를 고려하지 않고 단일 접속하여 실행한 결과이므로 동시접속자수가 늘어날 경우 실행 시간은 더 차이가 날수 있음을 알 수 있다.

[차트 1] SGDT와 SRSDM의 실행 시간 비교표



5 . 결 론

본 논문에서 제안한 시스템은 동적 문제 출제 시스템이 가지고 있는 문제점을 개선한 것으로 기존의 웹 서버에만 과부하를 주던 임의 추출 방식에서 탈피하여 데이터베이스 서버와 웹 서버 그리고 클라이언트에 골고루 임의 추출의 기능을 수행하게 함으로써 임의 추출에 의한 과부하를 최대한 줄이는 장점이 있다. 따라서 클라이언트에서는 기존의 시스템에 비하여 처리 속도가 약12.3% 개선되었다. 특히 여기서 제안한 시스템은 동시접속자가 많을 경우, 그 성능은 더욱 발휘된다.

향후 연구 과제로는 동적 문제의 다양화를 높일 수 있는 시스템으로 전환을 가져와야 한다. 이는 기존의 텍스트기반의 동적 문제에서 탈피하여, 다양한 동영상, 사운드, 음성 지원 등과 같은 멀티미디어적 문제 출제 시스템으로의 전환을 하여 최근의 청소년 경향에 맞추어가야 할 것이다.

참 고 문 헌

[1] 최돈은, 서현진, 박기석, 이재영, "동적인 문제출제 시스템의 설계 및 구현", 정보과학회, 학술발표논문집 제27권 1호 , 2000. 4

[2] 박기석, 김원진, 원대희, 이재영, "개인 능력 정보를 이용한 동적 문제 출제 시스템", 정보과학회, 학술발표논문집 제27권 2호 2000. 10

[3] 신재훈, 김종훈, "CGI를 이용한 교육용 실시간 대화 애플리케이션 구현", 한국정보과학회, 학술 발표논문집, 제27권 2호 2000

[4] 정원혁, "전문가를 위한 지름길 Microsoft SAL Server 7.0", 대림, 1999.9

[5] Francis, Fedordv, Harrison, Homer, Murphy, Sussman, Smith, Wood, "Professional Active Server Page 2.0", WROX Press,1999