대규모 온라인 검색 요구를 효율적으로 처리하기 위한 KRISTAL-II 웹 게이트웨이의 설계 및 구현

(Design and Implementation of the KRISTAL-II Web Gateway for Efficiently Processing a Large Number of On-line Retrieval Requests)

이 기 용 † 곽 태 영 † 서 정 현 †† 김 명 호 †††

(Ki Yong Lee)(Tae Yeong Kwak)(Jung Hyun Seo)(Myoung Ho Kim)

요 약 웹 게이트웨이는 WWW와 데이타베이스를 연동하는 인터페이스의 핵심을 차지하는 부분이다. KORDIC(Korea Research & Development Information Center)에서 개발된 기존의 KRISTAL-II 정보 검색 시스템[1]은 단순 CGI 구조의 웹 게이트웨이를 채택하고 있다. 이러한 구조의 웹 게이트웨이는 구현이 쉬우나 대규모의 검색 요구를 효율적으로 처리하기에는 부적합한 구조라는 단점이 있다. 웹을 통한 검색 요구가 점점 증가하는 추세임을 감안할 때, 대규모의 검색 요구를 효율적으로 처리할 수 있는 웹 게이트웨이를 개발하는 것은 매우 중요한 일이다. 본 논문에서는 기존 KRISTAL-II 웹 게이트웨이의 단순 CGI 구조를 대규모의 검색 요구의 효율적인 처리에 적합한 3-tier 클라이언트-서버 구조로 개선하였다. 실험을 통해 제안하는 새로운 웹 게이트웨이의 성능을 평가한다.

Abstract The web gateway is key technology for inter-operating WWW and databases. The previous KRISTAL-II information retrieval system, developed by KORDIC(Korea Research & Development Information Center), used a simple CGI structure web gateway. While the simple CGI structure web gateway is easy to implement, it is not suitable for processing a large number of on-line retrieval requests. When considering the growth of the Internet and WWW, it is very important to develop a web gateway efficiently supporting a large number of concurrent users. In this paper, we propose a 3-tier client-server structure web gateway for the KRISTAL-II information system. We also evaluate the performance of the proposed web gateway through experiments.

1. 서 론

인터넷과 웹(Web)이 대중화됨에 따라 많은 온라인 정보 검색 시스템들이 웹을 기반으로 하는 정보 검색 인터페이스를 제공하고 있다. 웹 정보 검색 시스템인 Yahoo[2]나 Lycos[3], 전산학 기술 문서 도서관인 NCSTRL[4]등은 이러한 시스템의 대표적인 예이다. 온라인 정보 검색 시스템들은 대부분 웹 인터페이스를 통해 데이타베이스에 접근하여 사용자의 질의를 처리하게 된다. 연구개발정보센터(Korea Research & Development Information Center)에서 개발한 KRISTAL-II 정보 검색 시스템[1]은 대용량의 데이타베이스로부터 사용자의 질의를 만족하는 문서들을 검색하는 시스템으로서, 다른 온라인 정보 검색 시스템들과 마찬가지로 웹 인터페이스를 통해 KRISTAL-II 검색 엔진에 접근하여 주어진 질의를 만족하는 문서들을 검색하게 된다. 웹을 통해 데이타베이스에 접근하기 위해서는 웹과 데이타베이스간의 연동 기술이 필수적이며, 웹 게이트웨이는 웹

† 비 회 원 : 한국과학기술원 전자전산학과
kylee@dbserver.kaist.ac.kr
tykwak@dbserver.kaist.ac.kr
†† 비 회 원 : 연구개발정보센터 연구원
jerry@kordic.re.kr
††† 종신회원 : 한국과학기술원 전자전산학과 교수
mhkim@dbserver.kaist.ac.kr
논문접수 : 2000년 1월 11일
심사완료 : 2000년 6월 7일

인터페이스와 데이타베이스를 연결해주는 웹-데이타베이스 연동 기술의 핵심을 차지하는 부분이다.

웹 게이트웨이는 다양한 형태로 구현이 가능하며, 구현 방법에 따라 크게 '검색 서버 확장 방법'과 '웹 클라이언트 확장 방법' 두 가지로 구분된다[5]. 검색 서버 확장 방법에는 CGI[6] 이용 방식, API 확장 방식, 전용 서버 방식 등이 있으며, 웹 클라이언트 확장 방법에는 외부 뷰어(viwer) 이용 방식과 웹 브라우저 확장 방식 등이 있다. 이 중 CGI 이용 방식은 다시 단순 CGI 구조의 단순 CGI 실행화일 방식과 3-tier 클라이언트-서버 구조의 응용 서버 방식으로 나눌 수 있다. 단순 CGI 실행화일 방식은 CGI를 통해 곧바로 데이타베이스 응용 프로그램을 수행하는 방식이며, 응용 서버 방식은 데이타베이스 응용 프로그램이 서버로 동작하고, CGI를 통해 이 응용 서버와 통신하는 클라이언트를 수행하는 방식이다. 응용 서버 방식에서는 CGI를 통해 수행된 클라이언트가 검색 인자를 응용 서버에게 전달하고, 다시 응용 서버로부터 검색 결과를 전달받아 이를 웹 서버에게 전달한다. 응용 서버 방식은 웹 서버-클라이언트-응용 서버의 3단계로 이루어진 3-tier 클라이언트-서버의 구조를 가진다. 현재 KRISTAL-II에서 제공되는 웹 게이트웨이는 단순 CGI 실행화일 방식을 채택하고 있다.

단순 CGI 실행화일 방식의 웹 게이트웨이는 데이타베이스 시스템이 제공하는 기존의 프로그램 라이브러리를 그대로 사용할 수 있고 구조가 간단하므로, 개발이 쉽고 개발 비용이 작다는 장점이 있다. 그러나 대규모의 온라인 검색 요청을 처리하는데 있어서 단순 CGI 실행화일 방식의 웹 게이트웨이는 다음과 같은 문제점을 가지고 있다.

- 매번 호출되어 실행되는 프로그램의 크기가 크다:

단순 CGI 실행화일 방식에서는 사용자의 요청이 들어올 때마다 요청을 처리하는데 필요한 모든 작업을 수행해야 하는 큰 크기의 프로그램이 실행되어야 한다. 실행되는 프로그램의 크기가 크면 프로그램을 메모리에 적재하는 시간이 길어지는 한편, 시스템의 가용 메모리 영역이 줄어들게 된다. 이로 인해 여러 프로그램이 동시에 수행되는 경우에는 가상 메모리 기법에 따라 디스크로의 접근이 빈번해지는 한편, 메모리 부족 현상이 나타나게 된다. 이는 프로그램의 수행 속도 및 시스템 전체의 성능 저하를 가져온다.

- 데이타베이스 시스템과의 연결 설정 및 해제가 반복된다:

데이타베이스 시스템에 접근하는 응용 프로그램은 수행 초기에 데이타베이스 시스템과의 연결을 설정하는 등의 준비 작업을 하고 수행이 끝날 때 데이타베이스 시스템과의 연결을 해제하는 작업을 한다. 이 과정에서 데이타베이스 시스템은 많은 작업을 수행한다. 따라서 데이타베이스 응용 프로그램이 반복적으로 수행되어야 하는 단순 CGI 실행화일 방식에서는 이렇게 수행 비용이 많이 드는 작업이 반복적으로 수행되므로 시스템의 성능이 저하되게 된다.

3-tier 클라이언트-서버 구조를 갖는 응용 서버 방식의 웹 게이트웨이는 위의 문제들을 해결하기 위해 제안된 구조이다[5]. 이 구조에서는 데이타베이스와 연동하여 검색을 수행하는 서버 프로그램을 두어, 검색 요청이 들어올 때마다 CGI를 통해 클라이언트가 수행되어 서버에게 검색 요청을 전달하도록 한다. 이러한 구조의 웹 게이트웨이에서는 CGI를 통해 수행되는 클라이언트 프로그램의 크기가 작으므로 여러 개의 클라이언트 프로그램이 수행되더라도 메모리 부족 현상이 나타나지 않게 된다. 그리고, 한 번 설정된 데이타베이스 시스템과의 연결을 계속 이용하게 되므로 검색 요청이 들어올 때마다 데이타베이스 시스템과의 연결 및 해제를 반복적으로 수행할 필요가 없다.

응용 서버 방식의 웹 게이트웨이는 다른 방식의 웹 게이트들과는 달리 CGI와 데이타베이스 프로그램 라이브러리를 이용하여 구현될 수 있다. 따라서 웹 브라우저를 확장하거나 외부 뷰어를 구현해야 하는 웹 클라이언트 확장 방식에 비해 구현에 필요한 비용이 적다. 마찬가지로 전용 서버 방식이나 API 확장 방식과 같은 검색 서버 확장 방식에 비해서도 구현 비용이 적을 뿐만 아니라 범용성이 좋다고 할 수 있다.

본 논문에서는 대규모의 온라인 검색 요청을 효과적으로 처리할 수 있도록 새로 설계된 3-tier 클라이언트-서버 구조를 가지는 응용 서버 방식의 KRISTAL-II 웹 게이트웨이에 대해 설명한다. 본 논문의 구성은 다음과 같다. 2장에서는 기존의 웹 게이트웨이의 종류 및 구조에 대해 알아보고, 3장에서는 기존 KRISTAL-II 웹 게이트웨이의 문제점을 개선하기 위해 새롭게 설계 구현된 새로운 KRISTAL-II 웹 게이트웨이에 대해 설명한다. 4장에서는 실험을 통해 새로운 KRISTAL-II 웹 게이트웨이의 성능을 평가하고, 5장에서 결론을 맺는다.

2. 웹 게이트웨이의 종류 및 구조

웹 게이트웨이는 구현 방식에 따라 크게 서버 확장 방식과 클라이언트 확장 방식으로 나눌 수 있다. 서버 확장 방식은 단순 구조의 단순 CGI 실행화일 방식과 3-tier 클라이언트-서버 구조의 응용 서버 방식, 웹 서

버의 확장 API를 이용하는 확장 API 방식, 그리고 전용 서버 방식으로 구분할 수 있다. 클라이언트 확장 방식은 웹 브라우저의 확장 기능을 이용하여 브라우저를 확장하는 방식과 검색 내용을 주고 받을 수 있는 새로운 형태의 뷰어(viwer)를 사용하는 외부 뷰어 방식으로 구분할 수 있다[5]. 현재까지는 CGI 이용 방식이 웹 게이트웨이의 개발에 주로 이용되고 있으며, 다른 방법들은 범용성과 이에 따른 표준화 문제를 갖고 있어서 별로 이용되지 않고 있다.

(1) 단순 CGI 실행화일 방식

CGI를 통해 수행되는 프로그램(CGI 실행화일)을 통하여 직접 데이타베이스에 접근하는 방식이다. CGI 실행화일은 DBMS가 제공하는 프로그램 라이브러리를 이용하여 개발할 수 있으므로 개발이 매우 용이하다. 그러나 실행화일이 크기 때문에 프로그램 실행의 초기화 시간이 길어지며, 다수의 프로그램이 동시에 수행될 때는 시스템의 성능이 저하되고 데이타베이스와의 연결 설정 및 해제를 반복하는 단점을 가지고 있다.

(2) 응용 서버 방식

앞서 설명한 단순 CGI 실행화일 방식의 문제점을 해결하기 위해 데이타베이스를 접근하는 프로그램을 서버로 수행되도록 하는 방식이다. 이 경우 웹 게이트웨이는 웹 서버에 의해 수행되어 응용 서버와 웹 서버간의 통신을 담당하게 되는 클라이언트, 그리고 데이타베이스 검색 및 HTML 문서 생성 기능을 담당하는 응용 서버 및 웹 서버로 이루어진 3-tier 클라이언트-서버 구조를 가지게 된다.

(3) 확장 API 방식

웹 서버들 중에는 서버의 기능을 프로그래머가 확장할 수 있도록 API(Application Program Interface)를 제공하는 것들이 있다. 확장 API 방식은 웹 서버에서 제공하는 API를 사용하여 데이타베이스 응용 프로그램들을 수행하는 방식이다. 확장 API를 이용하여 구축된 응용 프로그램은 웹 서버의 수행 도중에 필요에 따라 동적으로 링크되어 웹 서버와 하나의 프로세스로 수행되는 것이 일반적이기 때문에 CGI를 이용하는 방식에 비해 속도가 빠르다는 장점이 있다. 그러나 웹 서버들이 제공하는 API의 표준이 존재하지 않으므로 웹 서버마다 다른 웹 게이트웨이를 개발해야 한다는 단점이 존재한다.

(4) 전용 서버 방식

일반적인 웹 서버는 사용자가 원하는 문서를 전달하고, 사용자가 원하는 프로그램을 CGI를 통해 수행하는 기능만을 수행한다. 전용 서버 방식은 일반적인 웹 서버를 이용하지 않고, 특정한 데이타베이스 시스템에 직접 접근할 수 있는 기능을 갖춘 확장된 기능의 웹 서버를 이용하는 방식이다. 이 방식은 웹 인터페이스를 통해 DBMS에 접근하기가 쉽다는 장점이 있으나 특정 DBMS에 국한된 기능을 제공한다는 단점이 있다.

(5) 외부 뷰어 방식

웹 브라우저는 자신이 보여줄 수 없는 문서들의 실현(presentation)을 위해 외부 응용 프로그램을 지정하고 호출할 수 있는 기능을 제공하기도 한다. 외부 뷰어 방식은 이러한 기능을 이용하여 웹 게이트웨이를 웹 브라우저의 외부 응용 프로그램으로 이용하는 방식이다. 웹이 가지는 제한점을 피해 데이타베이스를 접근하는 것이 가능하다는 장점을 지니고 있으나 개발에 드는 비용이 매우 크다는 단점이 있다.

(6) 브라우저 확장 방식

앞에서 언급한 외부 뷰어를 웹 브라우저의 확장 API를 이용하여 웹 브라우저와 결합하는 방식이 브라우저 확장 방식이다. 자바(Java)와 같은 언어로 구현된 웹 브라우저는 자바로 구현된 응용 프로그램과 손쉽게 결합될 수 있다. 그러나 이 방식은 브라우저가 확장 기능을 제공할 때에만 이용할 수 있고, 외부 뷰어 방식과 마찬가지로 개발 비용이 크다는 단점이 있다.

웹과 데이타베이스간의 연동을 위해, 1993년 Jason의 GSQL[7]을 시작으로 IBM의 DB2WWW[8], Oracle의 Oracle Application Server[9], Netscape의 NSAPI (NetScape Application Program Interface)[10]등을 포함하여 다수의 웹 게이트웨이가 개발되어 이용되고 있다. 연구개발정보센터의 정보 검색 엔진인 KRISTAL-II에서도 단순 CGI 실행화일 방식의 웹 게이트웨이를 제공하고 있다. GSQL과 DB2WWW는 단순 CGI 실행화일 방식을 사용하고 있으며 Oracle Application Server는 전용 서버 방식을 사용하고 있다. 또한, NSAPI는 확장 API 방식을 사용하고 있으며, [11]에서 개발된 UniSQL/X용 웹 게이트웨이인 UniWeb은 응용 서버 방식을 채택하고 있다.

3. 개선된 KRISTAL-II 웹 게이트웨이의 설계 및 구현

본 논문에서는 단순 CGI 실행화일 방식이 가지는 문제점을 해결하고, 대규모의 온라인 검색을 효과적으로 처리할 수 있도록 하기 위해 단순 CGI 실행화일 방식을 채택하고 있는 기존의 KRISTAL-II 웹 게이트웨이를 3-tier 클라이언트-서버 구조의 응용 서버 방식으로 개선하였다. 단일 서버가 검색 요청을 처리할 때는 서버

의 과부하 문제가 발생할 수 있으므로 이를 해결하기 위해 서버 클래스 개념을 제공하여 복수 개의 응용 서버가 수행될 수 있도록 하였다. 이를 위해 온라인 트랜잭션 처리 시스템에서 이용되는 구조 가운데 하나인 다중 대기행렬 다중 서버(multi-queue multi-server) 구조를 채택하였다. 또한 개발에 드는 비용을 줄이기 위해, 기존의 단순 CGI 실행화일 방식의 CGI 프로그램을 가능한 한 재사용하는 것을 목표로 하였다.

3.1 기존 KRISTAL-II 웹 게이트웨이의 구조

단순 CGI 실행화일 방식인 기존의 KRISTAL-II 웹 게이트웨이는 다음과 같은 다섯 개의 CGI 프로그램으로 구성되어 있다.

- retrieve.cgi : 주어진 검색어에 따라 문서를 검색하는 CGI 프로그램
- showabs.cgi: 검색된 문서의 요약을 출력하는 CGI 프로그램
- showcontent.cgi: 지정된 문서의 내용을 출력하는 CGI 프로그램
- look.cgi: 데이타베이스에 관한 정보를 출력하는 CGI 프로그램
- lookretrieve.cgi: 색인어 데이타베이스에서 필요한 색인어를 추출하는 CGI 프로그램

이들 CGI 프로그램들은 웹 문서 내에서 <FORM ACTION="/cgi-bin/retrieve.cgi" METHOD="POST">와 같이 FORM 태그(tag)의 액션(action)으로 지정되어 사용자의 요청이 들어올 때마다 웹 서버에 의해 수행된다. 단순 CGI 실행화일 방식의 KRISTAL-II 웹 게이트웨이가 동작하는 과정은 그림 1과 같다.

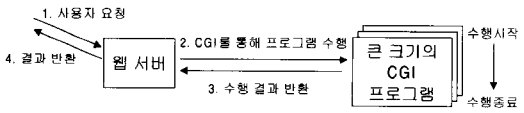

그림 1 기존 KRISTAL-II 웹 게이트웨이의 동작 구조

기존의 KRISTAL-II 웹 게이트웨이는 단순 CGI 실행화일 방식의 문제점을 그대로 가지고 있다. 즉, 매번 호출되어 실행되는 프로그램의 크기가 크다는 문제와 데이타베이스 시스템과의 연결 설정과 같이 프로그램의 초기에 수행되어야 하는 루틴이 반복된다는 문제를 가지고 있다.

3.2 새로운 KRISTAL-II 웹 게이트웨이의 설계

단순 CGI 실행화일 방식이 가지는 문제점을 해결하기 위해 설계된 응용 서버 방식의 새로운 KRISTAL-II 웹 게이트웨이는 다음과 같은 특징을 가진다.

(1) 3-tier 클라이언트-서버 구조 채택

사용자의 요구가 들어올 때마다 CGI를 통해 수행되는 프로그램은 검색 인자를 응용 서버에게 전달하고 그 결과를 웹 서버를 통해 사용자에게 전달하는 단순한 기능만을 수행하는 작은 프로그램이 되도록 한다. 이렇게 함으로써 매번 큰 크기의 CGI 프로그램이 수행됨으로 인해 발생되는 문제점들을 해결한다. 또한, 검색을 담당하는 응용 서버는 검색 요청의 수와 상관없이 일정한 수만큼을 가지도록 하며 데몬(daemon)의 형태로 항상 수행되는 상태가 되도록 한다. 이로써 검색 요청이 들어올 때마다 데이타베이스와의 연결 설정 및 해제 등의 작업이 반복적으로 수행되는 문제를 해결한다. 전체적인 시스템은 웹 서버-클라이언트-응용 서버 3단계의 계층을 가지는 3-tier 클라이언트-서버 구조를 가지게 된다.

(2) 서버 클래스 개념 제공

다중 클라이언트-단일 서버 구조의 시스템에서는 클라이언트의 수가 너무 많아지면 서버의 과부하 문제가 발생하게 된다. 이러한 문제는 응용 서버 방식의 웹 게이트웨이 시스템에서도 발생할 수 있다. 따라서 다중 서버 프로세스를 이용하여, 다수의 동일한 서버가 여러 클라이언트의 검색 요청을 동시에 처리할 수 있도록 하는 서버 클래스 개념을 제공한다. 이를 위해, 서버의 부하를 측정하여 부하에 따라 서버의 수를 조절할 수 있게 하는 기능이 제공된다.

(3) 기존 CGI 프로그램의 재사용

프로그램의 개발 기간을 단축하고 개발에 드는 비용을 최소화하기 위해 이미 구현 및 검증이 완료된 기존의 단순 CGI 실행화일 방식의 CGI 프로그램을 가능한 한 재사용한다. 즉, 기존 CGI 프로그램의 검색 코드를 되도록 수정하는 일 없이 그대로 재사용함으로써 구현에 드는 비용을 줄인다. 이를 위해, 기존의 CGI 프로그램들에 큰 수정을 가하지 않고 단지 서버로서 동작할 수 있도록 하는 코드만을 추가함으로써 기존의 CGI 프로그램들이 곧바로 응용 서버로 사용될 수 있도록 한다.

3.3 새로운 KRISTAL-II 웹 게이트웨이의 구조

3.2에서의 고려사항을 바탕으로 설계된 새로운 웹 게이트웨이는 그림 2와 같은 구조를 가지게 된다. 웹 서버로 사용자의 요청이 들어오면, CGI를 통해 클라이언트 프로그램이 수행된다. 클라이언트는 웹 서버로부터 전달받은 요청 메시지를 대기행렬(queue)을 통해 서버에게 전달한다. 서버는 요청 메시지를 받아 검색을 수행한 뒤, 검색 결과를 소켓(socket)을 통해 클라이언트에게

전달한다. 클라이언트는 전달받은 결과를 웹 서버에게 넘기고 수행을 종료한다. 웹 서버는 전달받은 결과를 다시 사용자에게 전달한다. 응용 서버 방식의 새로운 KRISTAL-II 웹 게이트웨이에서 각각의 모듈이 수행하는 기능 및 수행 과정은 다음과 같다.

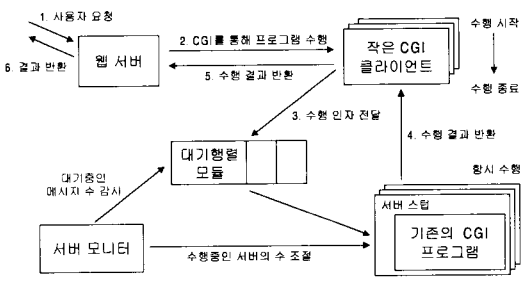

그림 2 KRISTAL-II의 변경된 웹 게이트웨이 구조

3.3.1 클라이언트 모듈

사용자의 요청이 들어올 때마다 웹 서버에 의해 CGI를 통해 수행이 시작된다. CGI를 통해 웹 서버에 의해 전달된 검색 관련 인자들을 받아 이를 대기행렬을 통해 응용 서버에게 전달하고, 응용 서버로부터 검색 결과를 전달받아 이를 웹 서버로 전달한다. 응용 서버로부터 전달받은 검색 결과는 표준 출력을 통해 출력됨으로써 웹 서버로 전달된다. 이러한 간단한 기능만을 수행하는 클라이언트 모듈은 기존의 CGI 프로그램에 비해 상당히 작은 크기를 가지게 된다. 그림 3은 클라이언트의 수행 과정을 나타낸다.

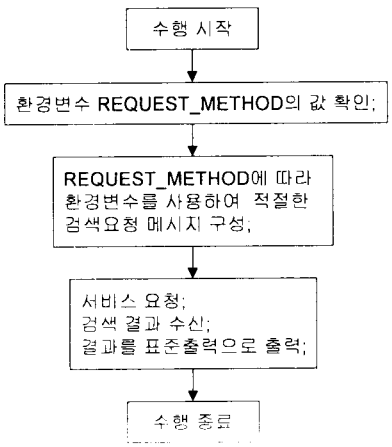

그림 3 클라이언트의 수행 과정

3.3.2 서버 스텁(stub) 모듈

기존의 CGI 프로그램들이 서버로 수행되기 위해 필요한 기능들을 담당한다. 여기에는 대기행렬을 통해 메시지를 전달받아 메시지를 해석하여 검색 인자를 분리하고, 검색 결과를 다시 클라이언트에게로 전달하는 기능이 포함된다. 기존의 CGI 프로그램은 printf()와 같은 함수들을 사용하여 검색 결과를 표준 출력을 통해 출력함으로써 이를 곧바로 웹 서버로 전달한다. 그러나 응용 서버 방식의 새로운 웹 게이트웨이에서는 검색 결과가 표준 출력대신 클라이언트에게로 전달되어야 한다. 기존 코드에 많은 수정을 가하지 않고도 기존 코드가 이러한 기능을 수행할 수 있도록 하기 위해서는 표준 출력을 가리키는 화일 디스크립터(descriptor)를 조작하는 루틴이 포함되어야 한다. 즉, 표준 출력을 가리키는 화일 디스크립터를 클라이언트와의 통신을 가리키는 디스크립터로 치환함으로써 표준 출력으로 출력되는 검색 결과들이 모두 클라이언트에게로 전달되도록 한다. 이렇게 함으로써 기존 코드에서 표준 출력으로 결과를 출력하는 모든 코드들은 아무런 수정을 가하지 않고서도 클라이언트로 결과를 보내는 코드로 사용될 수 있다. 그림 4는 서버의 수행 과정을 나타낸다.

한편 서버 스텁 모듈에서는 signal handler를 사용한다. 이 signal handler가 하는 일은, 서버에 종료를 알리는 signal이 들어왔을 때, 서버가 검색 요청 메시지를 처리하고 있는 중이라 할지라도 처리 중인 요청 메시지는 끝까지 마치고 종료하게 하는 일이다. signal handler는 종료를 알리는 signal이 들어오면 곧바로 서버를 종료시키는 대신, 서버가 현재 idle 상태인지 현재 검색 요청 메시지를 처리하고 있는 busy 상태인지를 판단한다. 그 후, 서버가 idle 상태이면 곧바로 서버를 종료하고 그렇지 않으면 그림 4의 While문의 Condition을 false로 변경시킨다. 서버는 검색 요청 메시지를 처리한

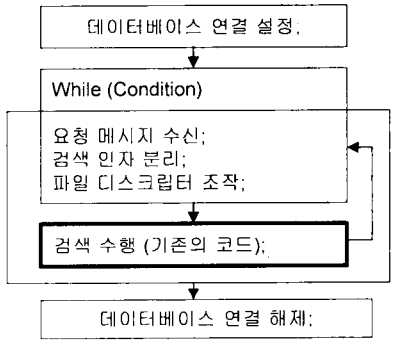

그림 4 서버의 수행 과정

뒤에 Condition을 보고 계속 수행할 것인지 아니면 종료할 것인지를 판단하게 된다.

3.3.3 서버 모니터 모듈

대기행렬에서 기다리는 검색 요청의 수를 시간에 따라 관찰하고 일정한 기준에 따라 서버의 수를 조절하는 기능을 수행한다. 또한, 전체 시스템의 초기화를 담당한다. 서버 모니터는 수행을 시작하게 되면 등록된 각 서비스에 대한 요청 메시지 대기행렬을 생성하고, 각 서비스를 처리하는 서버 프로세스를 지정된 수만큼 생성한다. 서버들이 수행되는 동안, 서버 모니터는 주기적으로 요청 메시지 대기행렬에서 처리를 기다리는 서비스 요청 메시지의 수를 검사하여 각 서버의 부하를 측정한다. 측정된 서버의 부하에 따라 서버 모니터는 서버의 수를 증가시키거나 감소시키게 된다. 그림 5는 서버 모니터의 동작 과정을 나타낸다.

서버 모니터는 그림 5의 while절의 N이 나타내는 시간 동안 대기행렬의 크기를 감시한다. N이 나타내는 시간이 지나면 측정된 대기행렬의 크기로부터 각 대기행렬의 로드를 구하고, 이에 따라 서버의 수를 조절한다.

그림 5에서의 "일정 수준"은 실험을 통하여 구할 수 있다. 실제 시스템에 들어올 요청 메시지의 빈도 수에 대해 서버의 수를 조절해가며 시스템의 성능을 측정하여 적절한 성능을 나타낼 때의 서버 수를 구한다. 이로부터 대략적으로 서버 하나가 처리해야 하는 요청 메시지의 수를 구할 수 있으며, 이로부터 "일정 수준"을 정할 수 있다.

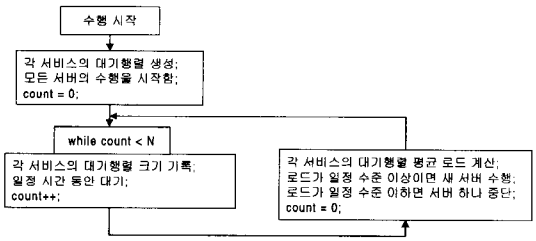

그림 5 서버 모니터의 동작 과정

3.3.4 클라이언트-서버 통신 모듈

웹 서버를 통해 수행되는 클라이언트와 응용 서버간의 통신은 여러 방법을 통해 구현될 수 있다. 클라이언트가 서버로 요청 메시지를 보낼 때는, 서버 모니터에서 각 서버의 부하를 측정할 수 있도록 하기 위해 온라인 트랜잭션 처리 시스템에서 사용되는 구조 가운데 하나인 대기행렬 구조를 사용하였다. 이러한 기능을 담당하는 대기행렬 모듈은 유닉스(Unix) 운영체제에서 제공하는 IPC(interprocess communication) 중의 하나인 메시지 대기행렬(message queue)을 사용하여 구현되었다. 클라이언트에서 서버로 보내는 요청 메시지는 다음과 같이 구성된다.

검색 요청 메시지 = (client_id, mesg_length, cgi_type, query_string, port)

- client_id : 검색 요청 메시지를 보내는 클라이언트의 프로세스 id
- mesg_length : 검색 요청 메시지의 길이
- cgi_type : 클라이언트가 호출된 방식 (POST 또는 GET)
- query_string : 검색인자
- port : 클라이언트가 서버로부터의 결과를 받을 소켓 port 번호

한편 응용 서버가 클라이언트에게로 결과를 보낼 때는, 비교적 긴 길이를 가진 결과를 빠르게 보낼 수 있으면서 화일 디스크립터의 조작만으로 기존 코드를 재사용할 수 있게 해주는 소켓(socket)을 사용하였다. 즉, 표준 출력을 가리키는 디스크립터를 소켓 디스크립터로 치환하면 기존 코드를 그대로 재사용할 수 있게된다. 그러나 응용 서버가 클라이언트로 결과를 보낼 때에는 소켓 외에도 몇 가지 선택 가능한 방법들이 있다. 4장에서는 소켓 외에 다른 통신 방법을 사용한 경우에, 각 시스템의 성능을 비교 평가한다. 그림 6은 클라이언트-서버 간의 통신을 나타내는 그림이다.

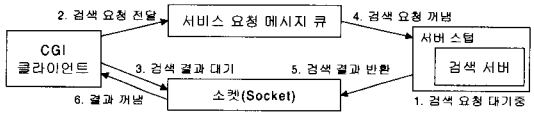

그림 6 클라이언트-서버 간의 통신

4. 성능 평가

단순 CGI 실행화일 방식의 문제점을 해결하기 위해 새로 설계된 응용 서버 방식의 KRISTAL-II 웹 게이트웨이의 성능을 평가하고자 몇 종류의 실험을 수행하였다. 대규모 검색 요청 환경을 실제로 구성하는 것은 매우 어려운 일이므로 본 논문에서는 많은 수의 검색 요청이 들어오는 환경을 시뮬레이션하는 방식을 통해 실험을 수행하였다.

4.1 실험 환경 및 방법

실험은 1018MB의 메모리에 8개의 CPU를 가지고 있는 SuperSPARC 시스템에서 이루어졌다. 여러 사용자

들이 여러 곳의 원격 사이트에서 동시에 대규모의 검색 요청 메시지를 보내는 상황을 실제로 구성하는 것은 현실적으로 어려운 일이므로, 대규모의 검색 요청 메시지가 들어오는 상황을 테스트 프로그램을 통한 시뮬레이션으로 대신하였다. 실제 상황에서는 사용자의 요구가 들어오면 그 때마다 웹 서버가 지정된 프로그램을 수행시키게 된다. 그러나 시뮬레이션을 통한 모의 실험에서는 웹 서버 대신에 실험을 위해 작성된 테스트용 프로그램이 대신 프로그램을 수행시키도록 하였다. 이 테스트 프로그램은 매개변수로 요청 메시지의 개수 및 평균 도착 시간을 받아 요청 메시지를 발생시키게 된다. 요청 메시지가 발생되면 단순 CGI 실행화일 방식의 기존 시스템의 경우에는 기존의 CGI 프로그램이, 응용 서버 방식의 새로 구현된 시스템의 경우에는 클라이언트가 테스트 프로그램에 의해 수행을 시작한다.

수행 성능은 초당 발생되는 정해진 개수의 요청 메시지에 대해, 이들을 모두 처리하는데 걸리는 시간으로 평가하였다. 즉, 1초에 10개, 20개 또는 100개의 요청 메시지가 들어온다고 하였을 때, 이들 요청 메시지를 모두 처리하는데 걸리는 시간을 측정하여 이를 성능 평가의 척도로 하였다.

4.2 실험 결과

3.3.4에서 언급한 바와 같이, 응용 서버에서 클라이언트에게로 검색 결과를 보낼 때에는 여러 종류의 통신 방법을 사용할 수 있다. 이 절에서는 기본적으로 소켓을 사용하는 경우 외에도 다른 통신 방법을 사용하는 경우에 대해 새로운 웹 게이트웨이 시스템과 기존 시스템의 성능을 비교 평가한다.

(1) 메시지 화일을 사용한 경우

메시지 파일 방법은 응용 서버가 비교적 큰 크기를 가지는 검색 결과를 모두 클라이언트에게 전송하는 대신에, 검색 결과를 담은 화일의 이름만을 클라이언트에게 전달하는 방법이다. 이 경우, 응용 서버는 검색 결과를 임시 화일에 저장한 뒤 이 화일 이름만을 클라이언트에게 전달하고, 클라이언트는 전달받은 화일 이름의 화일을 열어 그 내용을 표준 출력으로 출력하게 된다. 이 방법은 응용 서버와 클라이언트간의 통신에서 전달되는 메시지의 크기를 줄일 수 있다는 장점이 있다.

그림 7은 메시지 화일을 사용하는 응용 서버 방식의 새 시스템과 단순 CGI 실행화일 방식의 기존 시스템의 성능을 비교하는 그림이다. 초당 75개 이상의 요청 메시지가 들어오는 경우, 기존 시스템은 이들을 모두 처리하지 못하고 요청 메시지를 처리하기 위해 새로이 수행을 시작하는 프로세스들 중 일부를 부분적으로 kill하는 현

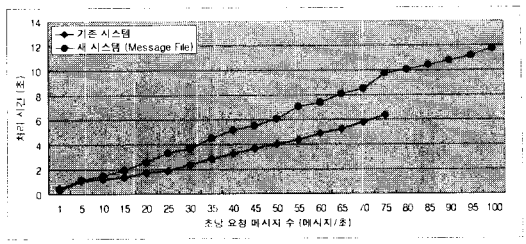

그림 7 기존 시스템과 메시지 화일을 사용하는 새 시스템의 성능 비교

상이 관찰되었다. 그림 7에서 기존 시스템의 처리 시간을 나타내는 그래프가 나타나지 않는 부분이 이를 의미한다. 이는 큰 크기를 가진 CGI 프로그램이 동시에 여러 개가 수행되어 메모리 등의 시스템 자원이 부족해짐으로써 발생되는 현상이라고 판단된다. 그러나 새로운 시스템은 그 이상의 비율로 요청 메시지가 들어오더라도 프로세스를 kill하는 일 없이 모든 요청 메시지를 잘 처리하고 있음을 알 수 있다. 이는 큰 크기를 가진 CGI 프로그램 대신에 작은 크기를 가진 클라이언트가 대신 수행되기 때문이다. 새로운 시스템은 초당 약 500개 정도의 요청 메시지가 들어오는 경우에도 프로세서를 kill하는 일 없이 이들을 모두 처리하였다.

그러나 그림 7에서 볼 수 있듯이, 새로운 시스템은 기존 시스템에 비해 속도의 측면에서 더 나쁜 성능을 보이고 있음을 알 수 있다. 이는 새로운 시스템의 구조로 얻을 수 있는 이득보다 메시지 화일 방식에 따른 반복되는 화일 I/O에 의한 오버헤드가 더 크기 때문이라고 판단된다.

(2) FIFO를 사용한 경우

그림 8은 새 시스템에서 서버가 클라이언트에게로 결과를 보낼 때 메시지 화일 방식 대신에 IPC 중의 하나인 FIFO(named pipe)를 사용하는 경우를 나타낸다. 이 경우, 클라이언트는 응용 서버로 요청 메시지를 보내면서 동시에 검색 결과를 받을 FIFO를 생성한다. 서버는 요청 메시지를 받은 후, 표준 출력 대신에 클라이언트가 결과를 기다리는 FIFO로 검색 결과를 출력하게 된다. 이 경우에서도 새로운 시스템은 초당 75개 이상의 요청 메시지에 대해서도 기존 시스템과 달리 프로세서를 kill하는 일 없이 모두 처리하고 있음을 볼 수 있다. 그러나 속도의 측면에서는 메시지 화일 방식보다는 좋은 성능을 보이고 있으나 기존 시스템에 비해서는 여전히 더 나쁜 성능을 보이고 있다. 이는 FIFO 방식에 따른 통신 오버헤드가 메시지 화일 방식에 따른 오버헤드에 비해 적기는 하지만 새로운 시스템의 구조로 인한 이득보다

는 여전히 크기 때문이라고 판단된다.

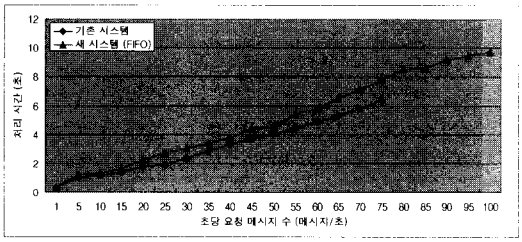

그림 8 기존 시스템과 FIFO를 사용하는 새 시스템의 성능 비교

(3) 소켓을 사용한 경우

그림 9는 소켓을 사용하는 응용 서버 방식의 새 웹 게이트웨이 시스템과 기존의 단순 CGI 실행화일 방식의 웹 게이트웨이 시스템의 성능을 비교하는 그림이다. 응용 서버가 클라이언트에게 검색 결과를 전달할 때 소켓을 사용하는 방식은 메시지 화일 방식이나 FIFO 방식에 비해 더 좋은 성능을 보이고 있음을 알 수 있다. 또한, 기존의 단순 CGI 실행화일 방식의 시스템에 비해서도 속도의 측면에서 더 좋은 성능을 보이고 있다. 물론, 메시지 화일 방식과 FIFO 방식의 경우와 마찬가지로 동시에 처리할 수 있는 요청 메시지의 수에 있어서도 기존 시스템에 비해 좋은 성능을 보이고 있다. 따라서 소켓을 사용하는 방법이 통신에 의한 오버헤드를 줄임으로써 기존의 단순 CGI 실행화일 방식의 시스템보다 더 빠른 성능을 가지는 한편, 동시에 더 많은 요청 메시지 수를 처리할 수 있는 방법임을 알 수 있다.

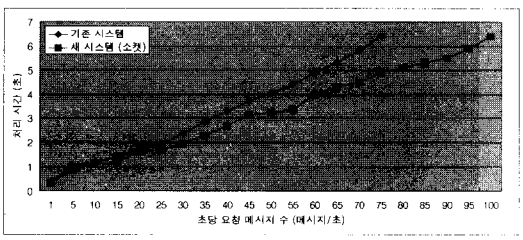

그림 9 기존 시스템과 소켓을 사용하는 새 시스템의 성능 비교

5. 결 론

본 논문에서는 웹-데이타베이스 연동에 있어서 핵심이 되는 웹 게이트웨이의 다양한 구조에 대해 고찰하고, 연구개발정보센터에서 개발한 정보 검색 엔진인 KRISTAL-II의 웹 게이트웨이를 대규모의 온라인 검색 요구를 효과적으로 처리할 수 있는 형태로 개선하였다.

본 논문에서는 단순 CGI 실행화일 방식의 기존의 KRISTAL-II 웹 게이트웨이를 3-tier 클라이언트-서버 구조에 기초를 둔 응용 서버 방식의 웹 게이트웨이로 개선함으로써, 단순 CGI 실행화일 방식의 기존 시스템이 가지는 단점들을 해결하였다. 또한, 기존의 CGI 프로그램들을 가능한 한 수정하지 않고 재사용하는 방법을 연구함으로써 개발에 드는 비용을 크게 감소시켰다.

성능 실험을 통해, 3-tier 클라이언트-서버 구조에 기반을 둔 응용 서버 방식의 새로운 시스템은 응용 서버와 클라이언트간의 통신 방법에 따라 다양한 성능을 보일 수 있음을 알 수 있었다. 이 중, 소켓을 사용하는 새로운 웹 게이트웨이 시스템은 기존의 단순 CGI 실행화일 방식의 시스템보다 더 빠른 처리 속도를 가지면서, 동시에 처리할 수 있는 요청 메시지의 수에 있어서도 기존의 시스템보다 더 좋은 성능을 가짐을 보였다. 이는 단순 CGI 실행화일 방식을 채택하고 있는 기존의 웹 게이트웨이를, 기존 코드를 크게 수정하지 않으면서 적은 개발 비용만을 가지고도 더 좋은 성능을 가진 응용 서버 방식의 웹 게이트웨이로 개선할 수 있음을 의미한다.

참 고 문 헌

[1] 연구개발정보센터, "정보검색을 위한 고성능 검색시스템 개발", 1996.
[2] Yahoo!, http://www.yahoo.com
[3] Lycos, http://www.lycos.com
[4] Networked Computer Science Technical Reference Library (NCSTRL), http://www.ncstrl.org
[5] 김평철, "A Taxonomy on Database Gateways for WWW," In Proceedings of the 3rd WWW Workshop, pp. 54-60, WWW-KR, March 1996.
[6] WWW Team, "CGI: Common Gateway Interface," CERN, http://www.w3.org/hypertext/WWW/CGI, 1994.
[7] Jason Ng, "GSQL-a mosaic SQL gateway," http://www.ncsa.uiuc.edu/SDG/People/jason/pub/gsql, 1993.
[8] Tam Nguyen and V. Srinivasan, "Accessing Relational Databases from the World Wide Web", Database Technology Institute IBM Santa Teresa Laboratory, 1996.
[9] Oracle Corporation, "The Integration Platform for Oracle Products and the Internet," Oracle White Paper, September 1998.
[10] NSAPI Programmer's Guide, http://developer.netscape.com/docs/manuals/enterprise/nsapi/index.htm
[11] 김평철, "UniWeb 2.0: A Gateway Between the

Web and Databases", 충남대학교, 1996.

이 기 용
1998년 한국과학기술원 전산학과 학사. 2000년 한국과학기술원 전산학과 석사. 2000년 ~ 현재 한국과학기술원 전산학과 박사과정 재학중. 관심분야는 데이터 웨어하우스, 무선 통신

곽 태 영
1993년 한국과학기술원 전산학과 학사. 1996년 한국과학기술원 전산학과 석사. 1996년 ~ 현재 한국과학기술원 전산학과 박사과정 재학중. 관심분야는 동시성 제어 및 회복 기법, 다차원 색인기법, Image Database.

서 정 현
1987년 한양대학교 수학과 학사. 1987년 12월 ~ 1993년 4월 시스템공학연구소 연구원. 1993년 4월 ~ 현재 연구개발정보센터 선임연구원. 관심분야는 정보검색, 자연어 처리.

김 명 호
정보과학회논문지 : 컴퓨팅의 실제
제 6 권 제 3 호 참조