논문 99-24-3B-8 한국통신학회논문지 '99-3 Vol.24 No.3B

Java CGI에 의한 이종정보시스템의 웹통합 연구

정회원 임 인 택*, 백 승 구*, 임 경 수**, 김 수 정***, 김 종 근*

A study on Integration of Heterogeneous Network Information Systems into Web by Java CGI

In-Taek Leem*, Sung-Gu Back*, Kyung-Soo Lim**, Soo-Jeong Kim***, Chong-Gun Kim*
Regular Members

요 약

독립적으로 개발되어 사용중인 이종 정보처리 시스템들을 웹 환경으로 통합하기 위한 효율적인 설계 방법론들이 연구되고 있다. 이러한 연구들은 웹의 편리한 사용자 인터페이스, 저비용, 확장성, 폭 넓은 사용자층의 매력을 바탕으로 활용범위를 넓혀 나가고 있다. 하지만 사용자 요구의 다양화와 데이터 양의 급격한 증가로 인해 웹서버의 부하는 점점 증가하고 있다. 이러한 환경에서 사용자들에게 효율적인 서비스를 제공하기 위해 서버의 부하를 줄이기 위한 기술 개발이 요구되고 있다. 다양한 플랫폼의 데이터베이스를 웹을 통해 서비스 받을 경우 클라이언트의 데이터베이스 검색 요청은 웹서버의 CGI를 통해 처리해왔으나 웹서버에 대한 부하의 집중으로 인해 클라이언트의 수가 많을수록 성능이 급격히 저하되는 현상이 발생한다. 이에 대한 개선방법으로 클라이언트에서 직접 데이터베이스 검색을 가능하게 하는 자바 CGI를 이용하는 방법 등을 통해 웹서버에 집중되는 부하를 효과적으로 분산시키는 방법을 적용한다. 이렇게 이종 정보처리 시스템을 웹으로 통합할 경우 CGI를 이용한 웹서비스 구조와 자바를 이용한 웹서비스 구조의 성능평가 모델을 제안하고, 이를 시뮬레이션을 통해 성능 평가한다.

ABSTRACT

Heterogeneous Network Information Systems(HNIS) are independently developed. Some design methods which efficiently integrate the HNISs into Web environments are studied. Those studies which are based on convenient user interfaces, lower cost, expansibility, and lots of users have broaden the range of using Web. However, because of rapid increase of various user requirements and data, the loads of Web Server Systems become gradually higher than before. This situations need technologies which reduce the loads of Web Server Systems. We derive Java CGI to integrate NHISs into Web environments. We propose a performance evaluation model for simulations to compare the performance of traditional CGI model with that of Java CGI model under various loads. Java CGI is a solution of the performance problems in overloaded CGI. Because Java CGI can efficiently distribute the loads of server to client.

I. 서 론

네트워크 상에 분산된 이종 정보자원은 보다 쉽고, 안정되고 일관된 방법으로 서비스를 제공하여야 한다. 이것은 사용자 인터페이스 측면에서 사용하기 쉬워야하며, 사용자의 요구 사항에 적절히 원하는 정보를 제공할 수 있는 기능들을 제공하여야 한다. 또한 시스템 측면에서는 기존의 시스템들과 쉽게

* 대구미래대학 멀티미디어 정보과학과 전임강사
** 연암공업대학 컴퓨터정보기술과 *** 경동전문대학 사무자동화과
논문번호 : 98270-0629, 접수일자 : 1998년 6월 29일
※ 이 논문은 1997년 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음.

통합되거나 통합할 수 있는 인터페이스를 지원해야 하며, 기존의 시스템보다 빠르거나 최소한 동등한 성능을 보장하여야 한다. 그리고 시스템과 정보에 대한 접근이 통일된 방법으로 관리되어야 한다. 이러한 요구사항들을 만족하는 시스템은 인터넷 서비스인 웹을 이용하여 효과적으로 구축될 수 있다.

웹은 편리한 사용자 인터페이스와 멀티미디어 환경을 제공함으로써 폭발적으로 사용자가 늘어나고 있다[12, 13]. 다양한 용도로 사용할 수 있는 웹은 모듈 인터페이스 기술을 이용하여 동적이고 다양한 서비스[6]를 가능하게 해준다. 웹과 정보 시스템 혹은 웹과 데이터베이스 시스템과의 연동[2,4,7,11]을 위한 API(application programming interface)나 웹과의 신뢰할만한 인터페이스를 별도로 제공하지 않는 legacy 애플리케이션[9]에 대해서도 이들을 효과적으로 통합할 수 있는 모듈 인터페이스 방법을 웹은 제공한다. 이렇게 모듈 인터페이스는 다양한 장점을 갖는 웹서비스의 확장을 효과적으로 지원해준다. 이러한 이종 정보시스템과의 연동을 위한 대표적인 모듈 인터페이스에는 CGI(common gateway interface)[10] 방식이 있다.

본 연구에서는 모듈 인터페이스에 의한 이종 정보시스템을 웹으로 통합할 경우, 특히 데이터베이스 시스템을 웹으로 통합[11]하는 경우 기존 CGI를 이용한 웹서비스 구조와 자바(java) CGI를 이용한 웹서비스 구조를 비교 분석하고, 사용자 수의 증가에 따른 두 방식의 성능 평가를 수행한다. 본 논문의 구성은 다음과 같다. 2장에서는 웹의 관련 기술에 대해서 설명하고, 제3장에서는 이종 정보시스템의 웹통합 기술, 4장에서는 CGI 방식과 자바 방식을 이용한 시스템 통합의 성능 평가를 위한 모델을 제안한다. 그리고 5장에서는 성능 평가 결과 및 분석, 마지막 장에는 결론을 맺는다.

Ⅱ. WWW(World Wide Web)의 관련 기술

WWW(이하, 웹으로 칭함)은 1989년에 CERN (conseil europeen pour la recherche nucleaire)의 Tim Berners-Lee에 의해 제안된 하이퍼텍스트 개념에 기반을 가지는 분산형 정보시스템이다. 웹은 HTTP(hypertext transfer protocol)에 기반을 두고 있으며 기본적으로 HTML(hypertext markup language)로 기술되어진 문서를 주고받는 인터넷 서비스이다. 웹은 일반적으로 하나의 서비스 화면에 하이퍼링크(hyperlink)되어 있는 요소를 가지며, 이 요소가 HTML문서일 경우 선택하면 해당 문서를 다시 불러오게 된다. 웹의 중요한 특징은 다음과 같다.

- 인터넷에 산재해 있는 정보를 하이퍼텍스트 개념으로 연결한다.
- 하이퍼텍스트 정보를 취급하기 위한 수단을 제공한다.
- 다양한 자원을 접속하기 위해 통일된 접근 수단을 제공한다.

2.1 CGI

CGI는 웹서버와 서버 측의 응용 프로그램 사이를 연결하기 위한 일종의 게이트웨이 표준이다. 이는 웹서버가 일방적으로 서비스를 제공하는 데 그치는 것이 아니라, 사용자의 요구를 동적으로 수용할 수 있도록 하기 위해 만들어진 것이다. 클라이언트가 HTTP를 통해 게이트웨이 프로그램에 접근하면 서버가 그 프로그램을 동작시키고, 클라이언트로부터 보내진 데이터를 게이트웨이 프로그램으로 전달한다. 게이트웨이 프로그램은 들어온 데이터 처리를 마치면 결과를 서버 측으로 보내며 서버는 이를 클라이언트 측으로 전달한다. 즉, CGI는 웹서버와 외부 또는 게이트웨이 프로그램간의 데이터들의 교환방법을 기술한다. 게이트웨이 프로그램은 C, C++, Pascal과 같은 언어로 작성되어 컴파일되거나, Perl Script, TCL, 혹은 다양한 쉘(shell)프로그램과 같은 스크립트 언어로 작성할 수 있다. 그림 1은 CGI 실행 과정을 나타낸다.

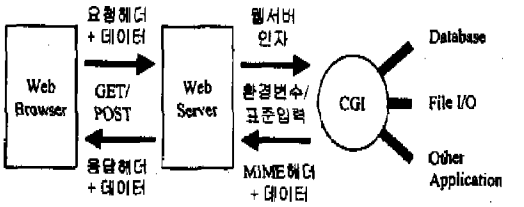


그림 1. CGI 실행 과정
Fig. 1 CGI execution procedure

웹브라우저에서는 URL(uniform resource locators)이나 폼에 입력된 값을 실제 웹서버에 전달할 때 요청헤더(request header)를 발생시켜 웹서버에 전달한다. 웹서버는 요청이 자체내의 HTML 문서인지 CGI 프로그램의 실행인지 분석하고, 이때 CGI 프로그램 실행이라면 웹서버는 CGI 프로그램 실행을 위해 별도의 프로세스를 생성시켜 실행한다. 웹서버로부터 전달받은 인자를 지정된 형식으로 디코딩(decoding)

한 다음 디코딩한 결과를 가지고 실제 원하는 처리를 수행한다. CGI 프로그램은 처리된 결과를 MIME 헤더와 함께 웹서버로 전달하고, 웹서버는 CGI의 MIME에 따라 적절한 응답헤더(response header)를 생성시켜 웹브라우저에 전달한다.

2.2 자바

자바는 1991년 Sun Microsystems에서 개발이 시작되어 1993년에 웹에 적용하기로 결정하고, 1995년 SunWorld 컨퍼런스에 자바를 공식 발표한 객체지향(object-oriented) 프로그래밍 언어와 환경이다[8]. HTTP와 같은 TCP/IP 네트워크 환경에서 작동하는 많은 프로토콜을 지원하는 라이브러리를 가지고 있고, URL을 이용하여 원격지의 컴퓨터에 있는 객체를 조작할 수 있다.

자바는 객체지향 언어인 C++과 비슷한 구조이지만, C++에서 잘 사용하지 않거나 모호한 기능을 제외시켜 코드를 단순화한 객체지향 언어이다. 또한 컴파일시 엄격한 데이터형을 검사함으로써, 프로그램 실행시 발생할 수 있는 비정상적인 상황 등을 미리 막을 수 있는 신뢰성과 안정성이 있다. 또한 자바 언어 수준에서 스레드를 만들 수 있는 기능을 제공하고 있을 뿐만 아니라, 그 자체도 멀티스레드(multi-thread) 기능을 가지고 있다. 또한 자바는 특정 플랫폼이 아닌 다양한 네트워크 환경과 하드웨어에서 작동할 수 있도록 바이트코드(bytecode) 개념[1, 14]을 도입했다.

바이트코드는 프로그램을 실행시키는 하드웨어와 운영체계에 관계없이 실행될 수 있게 하기 위해 제안된 포맷이다. 그림 2는 바이트코드의 실행을 보여주고 있다.

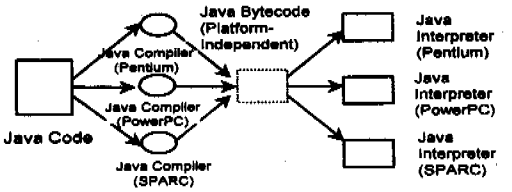


그림 2. 바이트코드 실행
Fig. 2 Bytecode execution

자바 애플릿(java applet)은 HTML 페이지에 포함되어 자바 가상기계 및 라이브러리를 탑재한 웹브라우저 상에서 실행되는 자바 프로그램을 말한다. 네트워크를 통해서 원거리에 있는 서버로부터 애플릿을 구성하는 자바 바이트코드를 전달받아 수행하는 구조이다. 이에 비해 자바 애플리케이션은 우리가 흔히 사용하는 프로그램과 같이 독립적으로 수행되는 자바 프로그램을 말하는 용어로 JDK(java development kit), JRE(java runtime environment)에 포함된 자바 실행환경을 사용하여 실행시킨다. 본 논문에서 자바 CGI 프로그램이란 자바로 구현한 CGI 기능의 애플릿을 말한다.

Ⅲ. 이종 정보시스템의 웹통합 기술

모듈 인터페이스를 통한 이종 정보시스템의 웹통합 기술은 구조적 측면과 분산적 측면으로 구분할 수 있다. 구조적 측면은 모듈 인터페이스가 어디에 존재하는가에 대한 측면이고, 분산적 측면은 모듈 인터페이스의 설계에 플랫폼에 독립적인 도구를 채택하였는가 하는 것이다.

먼저 구조적 측면에서 살펴보면 크게 서버사이드 확장 방식과 클라이언트확장 방식으로 나뉜다[2]. 서버사이드확장 방식의 경우 CGI이용 방식, 서버API 방식, 전용서버 방식으로 나눌 수 있다. CGI이용 방식은 기존의 웹서버가 제공하는 CGI 기능을 이용한 방식으로, 실행구조에 따라 CGI 실행 모듈방식과 CGI서버 방식으로 나뉜다. 서버API 방식은 웹서버에서 제공하는 서버API를 이용한 방식이고, 전용서버 방식은 특정 DBMS 접속 등의 기능을 내재하고 있는 웹서버를 이용한 방식이다. 클라이언트확장의 경우는 웹브라우저가 지원하는 외부뷰어 접속 기능을 이용한 외부뷰어 방식과, 브라우저 자체에 데이터베이스 접속 등과 같은 기능을 포함시키는 브라우저확장 방식으로 크게 구분할 수 있다.

분산적 측면에서 분류하면 CGI나 서버API를 이용한 중앙집중식 서버구조와 플랫폼에 독립적인 자바를 이용한 분산 구조[4]로 나눌 수 있다. 중앙집중식 서버구조는 CGI나 서버API를 이용한 각종 웹응용 프로그램들이 웹서버 상에서 실행된다. 하지만 자바를 이용한 분산 구조는 웹서버가 실행 가능한 바이트코드를 클라이언트에게 전송하여 웹응용의 실행이 클라이언트에서 이루어진다. 본 연구에서는 이종 정보시스템의 웹통합 기술을 분산적 측면을 중심으로 기술한다.

3.1 중앙집중식 서버구조

3.1.1 CGI 방식

동시에 많은 사용자들이 서비스를 요구할 경우, 게

이트웨이의 실행구조에 따라 하나의 질의를 처리하기 위한 데이터베이스(또는 DB) 연결 비용, 프로세스 관리비용이 달라지고, 이는 전체 웹 시스템의 성능에 큰 영향을 준다.

(1) CGI 실행모듈 방식

CGI 실행모듈 방식은 웹과 네트워크 정보시스템 혹은 데이터베이스 시스템과 통합하는 가장 단순한 방법으로, 기존의 웹서버, 웹브라우저, 그리고 웹 관련기술(예, URL, HTTP, HTML 등)을 변경없이 사용할 수 있고, 구현 및 시험이 간단하며, 차후 응용의 확장으로 인하여 사용자 서비스 요구사항이 확대될 경우, 각 서비스 모듈만을 확장함으로써 해결할 수 있다. 하지만 동시에 많은 요구가 수행될 경우, 각 요구에 대한 CGI 프로세스가 매 번 생성되고 그 때마다 데이터베이스 연결, 파일 I/O 등이 수행된다. 또한 프로세스의 생성 및 종료, 프로세스들 간의 통신 및 자료복사, 그리고 빈번한 프로세스 교체 등의 원인으로 웹서버 시스템의 자원 부족 현상이 쉽게 발생하여 전체 시스템의 성능을 저하시킨다. 그림 3은 CGI 실행모듈 방식을 보여 주고 있다.

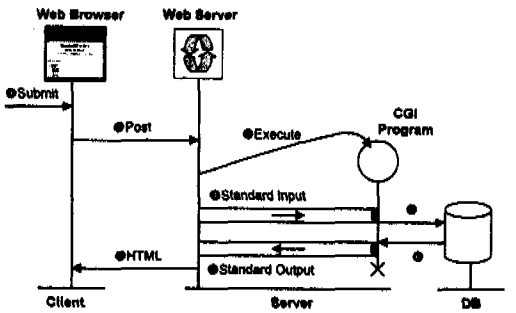


그림 3. CGI 실행모듈 방식
Fig. 3 CGI executable-module architecture

(2) CGI 서버 방식

CGI 서버 방식은 CGI 실행모듈 방식의 성능 문제를 해결하기 위해, CGI 프로세스가 데몬(demon)으로 동작한다. CGI 서버 방식은 웹서버의 CGI에 의해 구동되는 디스패처(dispatcher) 프로세스와 데이터베이스 연결 및 검색 등의 기능을 수행하는 서비스 모듈 데몬으로 구성된다. 웹서버의 CGI로 생성된 디스패처는 웹서버의 요구를 분석한 후, 그 요구를 처리할 수 있는 서비스 모듈 데몬을 찾아 전달한다. 서비스 모듈 데몬으로부터의 결과는 HTML 형식으로 디스패처에 전달되고, 디스패처는 그 결과를 웹서버에 전달한다. 하나의 디스패처 프로세스의 크기는 대략 수십 킬로바이트로 매우 작기 때문에 대규모 서비스 환경에서도 실행 중인 모든 디스패처 프로세스의 총 크기는 문제가 되지 않는다. CGI 실행모듈 방식과는 달리 서비스 모듈 데몬은 CGI 환경과 관계없이 구현될 수 있다. 서비스 모듈 데몬은 데이터베이스와 접속한 후 디스패처로부터 요구를 기다린다. 특히, 하나의 요구를 처리한 후에도 프로세스가 종료되지 않고 새로운 요구를 기다린다. 따라서, 이 방식은 DBMS 최적화 기술, 캐싱 등의 이점을 충분히 활용함으로써 웹서버 시스템의 성능을 크게 향상시킬 수 있다. 그림 4는 CGI 서버 방식의 실행구조를 나타낸다.

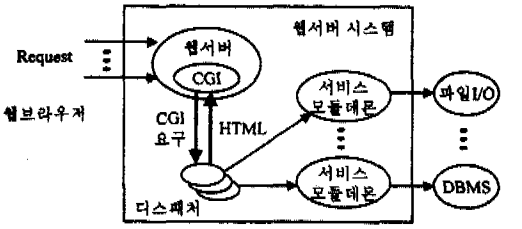


그림 4. CGI 서버 방식
Fig. 4 CGI server architecture

3.1.2 서버API 방식

CGI를 이용한 통합방식의 성능 문제를 해결하기 위해 웹서버 개발자들이 그들의 웹서버를 위한 API를 개발하였다. 이들 중 대표적인 것으로 Netscape사의 NSAPI(Netscape Server API), Microsoft사의 ISAPI(Internet Server API), IBM사의 ICAPI (IBM's Internet Connection API)가 있으며, 무료로 이용할 수 있는 Apache 서버도 또한 API를 제공하고 있다. 이것들은 웹서버의 기능을 응용 프로그래머가 확장할 수 있도록 서버API를 통해 웹서버로 하여금 데이터 베이스에 직접 접속할 수 있도록 하는 방식이다[3].

서버API를 이용하여 구축된 사용자 응용은 웹서버에 동적으로 링크되어 하나의 프로세스로 수행되는 것이 일반적이기 때문에, CGI 방식에 비해 성능이 우수하다. 또한 데이터베이스의 자료는 웹서버를 통해 전송되므로 프로세스간 자료 복사 및 프로세스 교체로 인한 부가 비용을 줄일 수 있다. 서버API를 이용하면 기존 CGI보다 많은 기능을 제공할 수 있는데, 액세스 제어, 로그 파일 액세스, 서버 요구 처리의 다른 상태로의 연결 등의 서버 내부 동작들의 행위를 수행할 수도 있다. 그림 5는 서버

API 방식을 나타낸다.

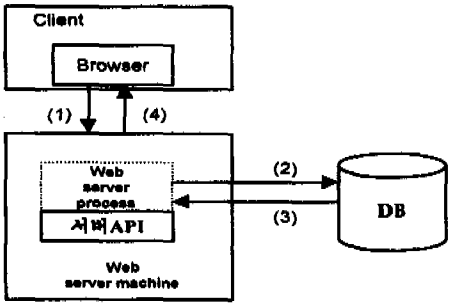


그림 5. 서버API 방식
Fig. 5 Server API architecture

하지만 서버 API 방식은 다음과 같은 문제점을 갖는다[15].

① 복잡성(complexity) : 서버 API는 구현 및 유지보수 비용이 증가한다.

② 언어 의존성(language dependence) : 애플리케이션들은 서버 API 의해 제공되는 언어(C/C++ 등)로 작성되어져야 한다. CGI 프로그램 작성을 위해 많이 쓰이는 Perl은 현존하는 어떤 서버 API와도 함께 사용될 수 없다.

③ 프로세스 고립성 결여 : 애플리케이션이 서버의 주소 공간에서 실행되기 때문에, 버그가 있거나 악의가 있는 애플리케이션이 서버를 붕괴시키거나 서버의 보안을 위협할 수 있다.

④ 독점성(proprietary) : 특별한 API로 작성된 애플리케이션은 그 개발자의 서버에만 한정될 수 있다.

이 중 서버 API의 가장 큰 단점은 표준화 부재로 인한 게이트웨이 코드의 제한적인 이식성(portability)이다.

3.2 자바를 이용한 분산 구조

3.2.1 자바를 이용한 2계층 방식

일반적인 웹서버는 데이터를 클라이언트에게 한번에 한번씩 전송하는 반면, 자바는 연속적으로 데이터를 클라이언트에게 보낼 수 있다. 즉, 자바는 데이타베이스와 연동시 연결 지향의 속성을 갖고 있기 때문에 필요할 때만 연결(non connection oriented)하는 HTML형식보다 더 효율적일 수 있다. 자바를 이용한 2계층 방식은 웹서버와 데이터베이스간의 정보교환이 아니라 클라이언트와 데이터베이스간의 직접적인 정보교환을 통한 효율적인 시스템 구성이 가능하다. 그림 6은 자바를 이용한 2계층 방식을 보여 주고 있다.

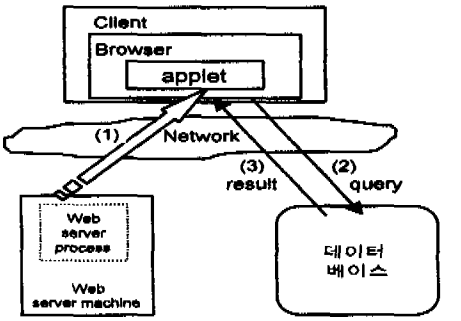


그림 6. 자바 2계층 방식
Fig. 6 2-tier architecture of java

3.2.2 자바를 이용한 3계층 방식

자바를 이용한 3계층 방식은 클라이언트와 데이터베이스 사이에 게이트웨이 서버를 통해서 보다 확대된 정보교환을 목적으로 하고 있다. 로컬 시스템 자원의 저장 혹은 삭제가 불가능한 애플릿의 단점을 극복하기 위해 애플릿과 데이터 베이스 사이의 게이트웨이 서버를 통해 데이터베이스 자원을 효율적으로 사용함(예, 캐싱 등)에 따라 대규모의 프로그래밍이 가능하다. 그림 7은 자바를 이용한 3계층 방식을 보여 주고 있다.

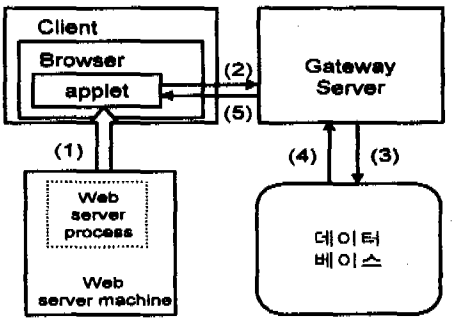


그림 7. 자바 3계층 방식
Fig. 7 3-tier architecture of java

Ⅳ. 평가 시스템 및 성능 평가 모델

웹의 성능에 영향을 주는 요소는 크게 3가지가 있다. 사용자의 요구를 웹서버에서 처리하는 과정, 이러한 요구를 데이타베이스에 전달하는 과정, 클라이언트와 서버의 네트워크 처리능력으로 나눌 수 있다. 사용자의 요구를 웹서버에서 처리할 경우 사용자의 웹서버 요청은 2가지로 분류할 수 있다. 먼저 이미지, 단순한 문서 파일인 HTML 문서와 같

은 정적파일(static file)을 요구하는 경우와, 데이터베이스나 다른 애플리케이션을 CGI로 연계한 동적 페이지(dynamic pages) 요구로 구분할 수 있다. 일반적으로 웹서버는 정적파일을 초당 100개를 처리할 수 있다면, 동적페이지는 10개정도를 처리하므로 동적페이지가 웹의 성능에 영향을 크게 미친다[3]. 동시에 많은 사용자가 웹을 통해 데이터베이스에 접속하거나 데이타베이스를 통한 다양한 요구 등도 웹의 성능에 영향을 미친다. 그리고 네트워크 환경 또는 서버나 클라이언트의 I/O 대역폭도 웹 성능에 영향을 미친다. 특히 사용자가 저속의 통신 선로(예, 14.4Kbps 또는 28.8Kbps 모뎀 등)로 웹서버에 정보를 요구할 경우 그 만큼 오랜 시간동안 접속을 유지해야하므로 웹서버의 접속 수가 증가하여 동시적 연결 병목(simultaneous connections bottleneck) 현상이 발생할 수 있다. 이것은 웹서버의 응답시간 지연 등의 성능에 영향을 줄 수 있다. 응답시간은 부하가 작을 때는 거의 차이가 나지 않지만, 시스템이 어떤 자원에 대한 병목을 유발할 때는 급속히 증가한다. 예를 들면, 서버측 애플리케이션이 초당 2Kbyte를 읽을 수 있는 클라이언트에 50Kbyte의 데이터를 전송한다면 총 25초가 소요된다. 이때 서버가 동시에 100개의 연결을 지원한다고 가정하면, 웹서버의 실제 처리율은 단지 초당 4(200Kbyte/sec)가 된다. 용량이 큰 파일이나 이미지를 주고받을 경우에도 오랜 시간동안 접속을 유지해야 하므로 성능에 영향을 미칠 수 있다.

본 연구에서는 웹의 성능에 영향을 미치는 요소 중 네트워크 환경은 단순화하고 웹서버 내에서의 CGI 처리, 데이터베이스에서의 처리만을 고려한다. 그리고 웹클라이언트 프로세스들은 웹서버와 동일한 LAN 환경에서 동작하게 함으로써, 네트워크 환경으로 인한 병목의 영향을 최소화하여 성능 평가의 결과에 대한 신뢰성을 보장한다.

일반적으로 사용하는 CGI 방식은 전형적인 중앙집중식 서버구조로서 많은 응답 처리 단계를 거쳐야 한다. 사용자가 질의를 웹서버에 보내면 서버는 입력된 데이터를 분석해서 에러가 있다면 웹서버를 거쳐 다시 웹브라우저에게 메시지를 보내고, 에러가 없다면 실행을 해서 결과를 웹서버를 거쳐 웹브라우저에게 보내게 된다. 이러한 구조의 문제점은 간단한 에러 처리조차 웹서버를 거쳐서 처리해야 하고, 동시에 많은 클라이언트의 서비스 요구시 병목현상을 일으킬 수 있다. 본 연구에서는 구현 및 성능 평가를 위해 기존의 CGI 실행모듈 방식과 자바를 이용한 2계층 CGI 방식만을 연구 대상으로 한다.

4.1 CGI 방식의 시스템 통합

4.1.1 CGI 방식에 의한 WWW-LINNET

CGI 방식에 의한 시스템 통합의 구현사례로서, CGI 실행모듈 방식과 CGI 서버 방식을 이용하여 영남대학교 도서관 도서 검색 시스템인 LINNET (library information network system) 시스템을 웹으로 통합하였다. 그림 8은 CGI 서버 방식을 이용한 WWW-LINNET 시스템의 구성이다.

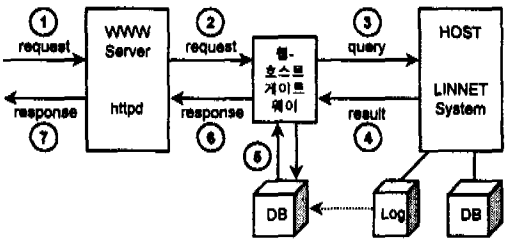


그림 8. WWW-LINNET 시스템
Fig. 8 WWW-LINNET system

그림 8에서 ① 웹클라이언트가 웹서버에게 질의를 보내면, ② 웹서버가 이러한 질의를 다시 웹-호스트 게이트웨이(web-to-host gateway)[5]로 넘겨준다. ③ 웹-호스트 게이트웨이는 사용자 요구마다 디스패쳐를 실행시킨다. 디스패쳐는 입력받은 데이터를 디코딩한 후, 해당되는 게이트웨이 서비스 모듈 데몬을 식별하여 사용자 입력을 전달한다. 해당 게이트웨이 서비스 모듈 데몬은 LINNET과의 통신을 위해 프로토콜 PDU(protocol data unit)를 생성한 후, LINNET과의 세션을 만들어 전송한다. ④ 수신한 PDU를 LINNET에서 처리하고 그 결과를 다시 게이트웨이 서비스 모듈 데몬 프로세스에게 전송한다. 이 때 데이타베이스 데이터와 Log 정보도 포함된다. ⑤ 게이트웨이 서비스 모듈 데몬이 전송된 결과를 자신의 로컬 데이타베이스에 반영하여 항상 호스트 측의 자료와 일관성을 유지한다. ⑥ 로컬 데이타베이스를 통한 사용자의 질의에 대한 응답은 서버에게 보내고, ⑦ 서버는 HTML 문서정보를 다시 웹클라이언트에게 전송한다.

4.1.2 CGI 방식의 성능 평가 모델

CGI 실행모듈 방식은 웹-호스트 게이트웨이의 프로세스 크기, 생성 및 종료, 프로세스간의 자료 복사 및 교체 등의 프로세스 관리 비용이 문제가 된

다. 특히 동시에 많은 요구가 수행될 경우, 각 요구에 대한 게이트웨이 프로세스가 매 번 생성되고, 그 때마다 데이터베이스와의 접속을 유발한다. 이것은 쉽게 시스템 자원 부족 현상을 발생시켜 전체 시스템의 성능을 저하시킨다. 그림 9는 이종 정보시스템의 웹통합을 위해 사용된 CGI 실행모듈 방식의 성능 평가 모델을 보여 주고 있다.

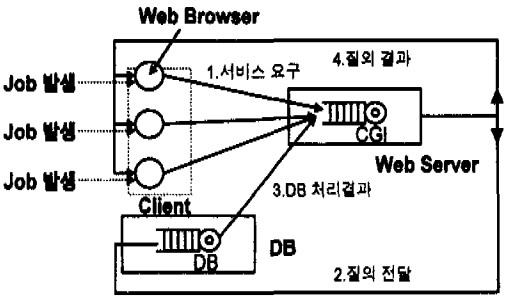


그림 9. CGI 실행모듈 방식의 성능 평가 모델
Fig 9. The model for CGI executable-module architecture

4.2 자바 방식의 시스템 통합

4.2.1 자바 2계층 방식을 통한 LINNET 시스템의 웹통합

플랫폼에 독립적인 특징으로 인해 분산적 구조를 가지고 있는 자바 방식은 연결 지향의 속성을 이용하여 웹클라이언트와 데이터베이스간의 효율적인 정보교환 모델을 제시한다. 웹클라이언트가 웹서버에게 특정 서비스를 요구하면, 웹서버는 자바 애플릿을 웹클라이언트에게 보내게 되고, 이 애플릿을 통해 웹클라이언트는 데이터베이스와 직접 통신을 하게 된다. 사용자의 질의를 애플릿이 분석하여, 만약 에러가 있다면 웹서버를 거치지 않고 애플릿에서 처리를 하고, 그렇지 않다면 데이터를 데이터베이스와 미리 정의된 프로토콜을 이용해 데이터베이스 시스템에 질의 데이터를 전달하게 된다. 그리고 처리 결과는 바로 클라이언트에게 전달하게 된다. 자바 애플릿을 이용하는 이러한 방법은 대규모 웹서비스 환경에서 다수의 웹클라이언트가 웹서버에게 서비스 요구시, CGI 기능의 애플릿을 클라이언트에게 보내므로 웹서버의 병목 현상 해소와 에러 처리를 웹클라이언트 내에서 처리 할 수 있어서, 웹서버의 과부하를 방지할 수 있다는 장점이 있다.

자바는 인터넷이나 다른 시스템에서 다운로드된 프로그램은 모두 외부 프로그램으로 간주하여 제한된 권한으로 실행되는 샌드박스(sandbox) 모델을 가지고 있다. 이러한 제한으로 자바 애플릿은 인터넷에서 다운로드 받은 호스트 외에는 접속을 할 수 없고, 로컬 파일도 쓸 수 없다. 자바 샌드박스 모델을 극복하기 위해서는 애플릿을 전자 서명하여 JAR 파일로 포맷하여 사용할 수 있다. 웹브라우저는 전자 서명이 된 애플릿이 신뢰된 자의 것이라고 인정되면 로컬 시스템의 프로그램과 같은 권한으로 실행한다. 그림 10은 자바를 이용한 WWW-LINNET의 실행과정을 보여준다.

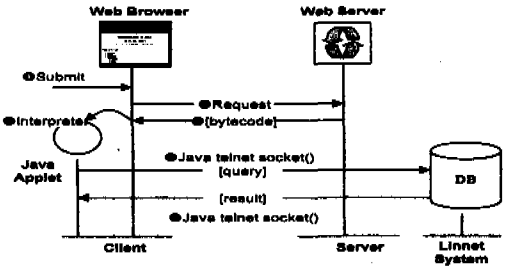


그림 10. 자바를 이용한 WWW-LINNET의 실행과정
Fig. 10 WWW-LINNET execution procedure using java

본 연구에서는 애플릿을 JDK1.1.5로 간단히 전자서명해서 사용할 수 있는 핫자바(HotJava) 브라우저 1.1.2를 사용한다. 그림 11은 자바 2계층 방식을 이용한 WWW-LINNET 시스템 구성이다.

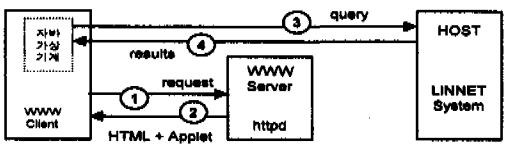


그림 11. 자바 2계층 CGI 방식을 이용한 WWW-LINNET
Fig. 11 WWW-LINNET using 2-tier architecture of java CGI

4.2.2 자바 방식의 성능 평가 모델

자바를 이용한 분산 구조는 웹서버와 데이터베이스간의 정보교환이 아니라 클라이언트와 데이터베이스간의 직접적인 정보교환을 통한 효율적인 시스템 구성이 가능하다. 그림 12는 자바를 이용한 성능 평가 모델을 보여준다.

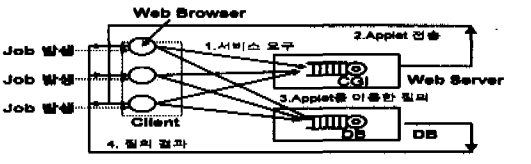


그림 12. 자바 방식의 성능 평가 모델
Fig. 12 The model for java architecture

4.3 CGI 모델과 자바 모델의 처리과정 비교

사용자가 웹브라우저를 통해 서비스 요구시 CGI 모델과 자바 모델은 그림 13과 같이 4단계의 과정을 거친다. CGI 모델의 경우 웹클라이언트의 요구는 웹서버를 거쳐 데이터베이스에게 전달되고, 처리 결과는 역방향으로 웹서버를 거쳐 웹클라이언트로 전달된다. 자바 모델에서는 웹클라이언트의 요구를 웹서버에서 처리하지 않고, 단지 웹서버에서는 자바의 바이트코드인 서비스 모듈을 클라이언트에게 전달한다. 바이트코드는 웹클라이언트의 자바 가상기계에 의해 인터프리트되어 실행코드로 변환되고, 데이터베이스에 직접 질의를 전달하고, 그 처리 결과를 다시 전달받는다. 서비스 처리 과정에서 두 모델의 큰 차이점은 중앙집중식 서버구조인 CGI 모델은 웹서버에서 2번의 처리를 한다. 반면에 분산 구조인 자바 모델은 웹서버에서 1번의 처리만을 수행한다. 그러므로 자바 모델은 웹서버가 고부하일 때, 웹서버의 부하를 웹클라이언트에게로 분산함으로써 CGI 모델보다 신뢰성과 안정성이 있다.

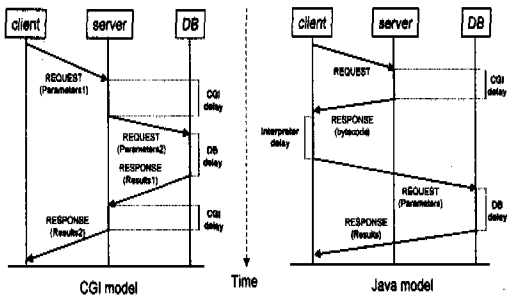


그림 13. CGI 모델과 자바 모델의 비교
Fig. 13 The comparison between CGI model and java model

성능 평가 모델의 파라메터로는 사용자의 요구를 처리하기 위해 웹서버에서 수행하는 CGI 프로세스의 처리시간과, 데이타베이스에서의 질의 처리시간, 자바 바이트코드가 클라이언트측에서 인터프리터되는 시간, 그리고 통신지연이 있다. 본 연구에서 모든 웹클라이언트와 웹서버가 동일 LAN 환경에서 동작하므로, CGI 모델과 자바 모델의 4단계 작업과정에서 통신지연은 거의 일정하다. 또한 두 모델에 있어서 동일한 질의를 사용하므로 데이터베이스에서의 질의 처리시간도 일정하다.

CGI 모델에서 첫 번째 CGI 지연은 CGI 프로세스 생성 및 초기화, 사용자 입력정보의 디코딩, 그리고 SQL 질의어 생성 등의 작업을 수행하는데 걸리는 시간이다. 두 번째 CGI 지연은 SQL 질의 수행 결과를 HTML로 변환하는데 걸리는 시간이다. 자바 모델에서의 CGI 지연은 CGI 프로세스 생성 및 초기화, 사용자 입력정보의 디코딩, 그리고 웹클라이언트로 바이트코드의 전송 등의 작업을 수행하는데 걸리는 시간이다. 자바 인터프리터 지연은 전송받은 바이트코드가 웹클라이언트의 자바 가상기계에 의해 인터프리트되어 실행코드로 변환하는데 걸리는 시간이다.

시뮬레이션 모델은 5개의 클라이언트에서 사용자 요구를 발생시키고, 각 클라이언트에서는 발생된 작업에 대한 응답이 돌아오면 작업전송간격에 따라 다음 작업이 계속 발생시킨다. 그리고 성능 평가를 위한 시뮬레이터로는 GPSS/H를 이용한다. 표 1은 두 모델의 성능 평가를 위한 시뮬레이션 파라메터를 나타내고 있다[1,3,8,16].

표 1. 시뮬레이션 파라메터
Table 1. Simulation parameters

	자바 인터프리터 지연	CGI 지연	데이타베이스 지연
CGI 모델		0.3, 0.2	0.5
자바 모델	0.6	0.3	0.5

V. 성능 평가 결과 및 분석

본 연구에서는 CGI와 자바의 성능 평가 모델을 기반으로 작업 전송 간격에 따른 평균 응답시간, 데이터베이스 이용률, 그리고 서버 이용률을 측정하였다. 여기서 작업 전송 간격은 웹클라이언트에서의 작업 발생수와 웹서버의 부하와는 반비례한다.

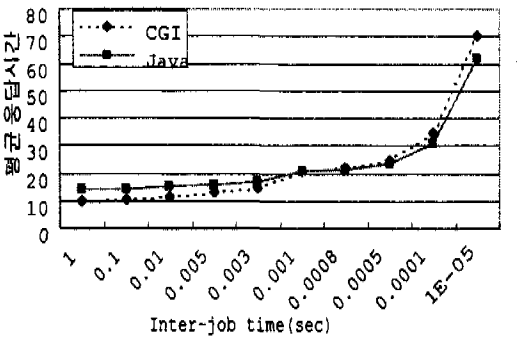


그림 14. 작업 전송 간격에 따른 평균 응답시간
Fig. 14 Average response time for inter-job time

그림 14는 작업 전송 간격에 따른 CGI 모델과 자바 모델의 평균 응답시간을 보여 주고 있다. 웹서버가 저부하일 때는 CGI 모델이 자바 모델에 비해 우수하다. 그러나 부하가 증가함에 따라 자바 모델이 CGI 모델에 비해 평균응답시간이 우수함을 보인다.

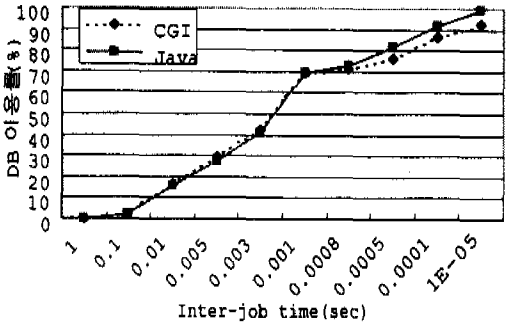


그림 15. 작업 전송 간격에 따른 데이터베이스 이용률
Fig. 15 Database utilization for inter-job time

그림 15는 작업 전송 간격에 따른 CGI 모델과 자바 모델의 데이타베이스 이용률을 보여 주고 있다. 웹서버가 저부하일 때, 두 모델의 성능 차이는 거의 나타나지 않지만, 부하가 점점 증가함에 따라 데이타베이스 이용률에 있어서 자바 모델이 우수함을 보인다.

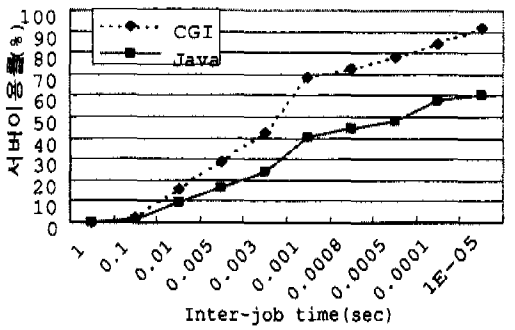


그림 16. 작업 전송 간격에 따른 서버 이용률
Fig. 16 Server utilization for inter-job time

그림 16은 작업 전송 간격에 따른 CGI 모델과 자바 모델의 서버 이용률을 보여주고 있다. 웹서버의 부하가 점점 증가함에 따라 CGI 모델이 자바 모델에 비해 서버 이용률이 급격하게 증가함을 보인다.

이와 같은 성능 평가 결과에서 웹서버가 고부하 상태로 갈수록, 많은 웹 자원을 사용하는 CGI 모델은 단위시간당 웹서버가 처리할 수 있는 사용자 요구 수와 이에 따른 데이터베이스와의 연결 수를 제한하였다. 하지만 자바 모델은 같은 상황에서 웹서버의 부하를 웹클라이언트에게 분산시킴으로써, 단위시간당 웹서버는 더 많은 사용자 요구 수와 데이터베이스와의 연결을 지원해서, 각 요구에 대한 평균 응답시간이 CGI 모델에 비해 우수하다는 것을 알 수 있다.

Ⅵ. 결 론

본 연구에서는 모듈 인터페이스에 의한 이종 정보시스템의 웹통합에서 CGI 방식과 자바 방식을 비교분석하였다. CGI 방식의 경우, 구현이 비교적 간단하다는 장점이 있지만, 동시에 많은 서비스 요구로 인한 웹서버 자원의 병목현상으로 전체 웹서버 성능에 문제를 일으킬 수 있다. 자바 방식의 경우, 처음 로딩시 인터프리터되는 시간이 많이 소요되므로 웹서비스에 문제가 있지만, 동시에 많은 사용자가 서비스를 요구할 경우, 서비스 모듈 자체를 클라이언트로 가져와서 실행하므로 CGI 방식에 비해 웹서버의 부하를 경감시키는 것으로 나타났다. 즉, 중앙집중식 서버구조의 CGI 방식과 분산 구조의 자바 CGI 방식은 부하의 증가에 따른 성능의 차이가 있음을 확인하였다.

추후 연구해야 할 과제로는 다양한 네트워크 환경에서의 성능평가와 시스템 부하에 따라 CGI 방식과 Java 방식을 지능적으로 혼합하여 부하를 분산시키고 성능을 향상시키는 방법이 제시되어야 할 것이다.

참 고 문 헌

[1] Todd A.Proebsting, Gregg Townsend Patrick Bridges, John H. Hartman, Tim Newsham, Scott A. Watterson, "Toba: Java For Applications A Way Ahead of Time(WAT) Compiler", COOTS'97, 1997.

[2] Pyung-Chul Kim, "A Taxonomy on the Architecture of Database Gateways for the Web", ICAST97/ICMIS97, 1997.

[3] Arun Iyengar, Ed MacNair and Thao Nguyen, "An Analysis of Web Server Performance", IEEE GLOBECOM'97, Vol.3, pp.1943-1947, 1997.

[4] Nick N.Duan, "Distributed Database Access

in a Corporate Environment Using Java", The Fifth International World Wide Web Conference, 1996.

[5] White Paper, Architecture Of Web-to-Host Gateways, [http://www.teubner.com/corridor/4.0/].

[6] 최용준, 임경수, 황도삼, 김종근, "계층적 정보 구조의 웹시스템 관리기술", 한국정보처리학회 논문지, 제5권 제5호, pp.1300-1310, 1998.

[7] Stathes P. Hadjiefthymiades, Drakoulis I. Martakos "Improving the performance of CGI compliant database gateways" The 6th International World Wide Web Conference, 1997.

[8] Robert Orfali, Dan Harkey, Client/server programming with Java and CORBA, John Wiley & Sons, Inc. 1997.

[9] Robert A. Barta, Manfred Hauswirth, "Interface-parasite Gateways", The Fourth International World Wide Web Conference, 1995.

[10] NCSA, The Common Gateway Interface,[http://hoohoo.ncsa.uiuc.edu/docs/cgi/overview.html].

[11] R. Eberhardt, C. Rueβ , C. Sinner, H. Scherand. Electronic Commerce-A Comparative Study of Web Based Database Access. ISS'97 : World Telecommunications Congress, pp. 97-104, Toronto, September 1997.

[12] M. F. Arlit and C. L. Williamson, "Web Server Workload Characterization: The Search for Invariants", Proc. of SIGMETRICS'96 ACM, May 1996.

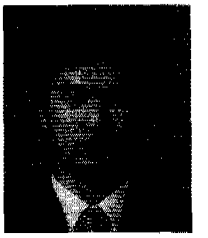
[13] T. T. Kwan, R. E. McGrath and D. A. Reed, "User Access Patterns to NCSA's World Wide Web Server", Technical Report, UIUCDCS-R-95-1934, Dept. of Computer Science, University of Illinois, 1995.

[14] Drew Dean, Edward W.Felten, Dan S. Wallach, "Java Security:From HotJava to Netscape and Beyond", IEEE Symposium on Security and Privacy, 1996.

[15] Open Market, Inc, FastCGI Technical White Paper, [http : // www.fastcgi.com / kit /doc/ fastcgi-white paper / fastcgi.htm].

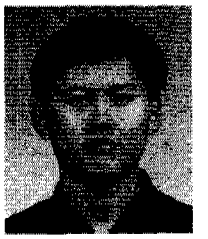
[16] System Performance Evaluation Cooperative (SPEC), SPECweb96 Benchmak, [http : // www.specbench.org / osg /web96 /].

임 인 택(In-Taek Leem) 정회원



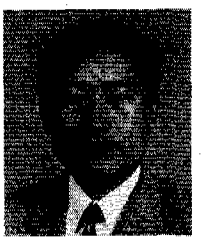
1996년 2월 : 영남대학교 건축공학과 졸업(공학사)
1998년 2월 : 영남대학교 대학원 멀티미디어정보통신학과 (공학석사)
1998년 3월~현재 : 영남대학교 대학원 컴퓨터공학과 박사과정
1998년 3월~현재 : 대구미래대학 멀티미디어 정보과학과 전임강사
<주관심 분야> 차세대 인터넷, 인터넷 응용기술, 시뮬레이션, 멀티미디어 통신
e-mail : itleem@white.yeungnam.ac.kr

백 승 구(Sung-Gu Back) 정회원



1997년 2월 : 경일대학교 제어계측공학과 졸업(공학사)
1998년 3월~현재 : 영남대학교 컴퓨터공학과 석사과정
<주관심 분야> 인터넷 응용기술, 네트워크 시스템, 자바 시스템
e-mail : sgback@nety.yeungnam.ac.kr

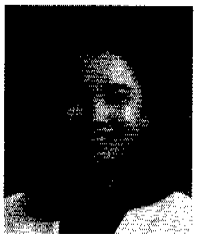
임 경 수(Kyung-Soo Lim) 정회원



1984년 2월 : 경북대학교 전자공학과 졸업(공학사)
1986년 2월 : 영남대학교 대학원 전자과 전자계산기 전공 졸업(공학석사)
1996년 2월 : 영남대학교 대학원 전자과 전자계산기 전공 졸업(공학박사)
1991년 3월~현재 : 연암공업대학 컴퓨터정보기술과

부교수
<주관심 분야> 분산처리, 컴퓨터망, 인터넷, 데이터베이스
e-mail : kslim@bjo.yonam-c.ac.kr

김 수 정(Soo-Jeong Kim) 정회원



1991년 2월 : 영남대학교 전산공학과 졸업(공학사)
1994년 2월 : 영남대학교 대학원 전산공학과 졸업 (공학석사)
1997년 2월 : 영남대학교 대학원 전산공학과 박사과정수료
1997년 3월~현재 : 경동정보대학 사무자동화과 전임강사
<주관심 분야> 컴퓨터 네트워크, 인터넷, 시뮬레이션
e-mail : sjkim@serve.kyungdong-c.ac.kr

김 종 근(Chong-Gun Kim) 정회원

1981년 2월 : 영남대학교 전자공학과 졸업(공학사)
1987년 2월 : 영남대학교 대학원 전자공학과 졸업 (공학석사)
1991년 3월 : 일본 東京電氣通信大學 졸업(공학박사)
1991년 3월~현재 : 영남대학교 컴퓨터공학과 부교수
<관심분야> 분산처리시스템, 분산 운영체제, 정보통신 기술, 인터넷 응용 기술, 시스템 성능평가
e-mail : cgkim@ynucc.yeungnam.ac.kr