웹 정보시스템의 서비스 성능 향상을 위한 부하균형 모델 제안

김 수 정† · 백 승 구†† · 김 종 근†††

요 약

본 논문은 이종정보처리 시스템들을 웹 환경으로 통합하는 경우에 사용자 부하를 분산시키는 방법을 연구한다. 서버지향 방식인 전통적인 CGI 모델과 클라이언트 지향 방식인 Java CGI 모델을 비교하고, 두 방식을 혼합하여 웹 서버의 부하상태에 따라 처리방식을 결정하는 LB(Load Balanced; 부하균형) CGI 방식을 제안한다. 세가지 방식에 대해 평가모델을 제시하고, 성능평가 결과 시스템의 부하나 서버의 성능에 상관없이 LB CGI 모델이 우수함을 컴퓨터 시뮬레이션을 통해 보인다.

A Load Balancing Model for Improving Performance of Web-Based Information System

Soo-Jeong Kim† · Sung-Gu Back†† · ChongGun Kim†††

ABSTRACT

In this paper, different methods of integrating heterogeneous network information systems into the Web are observed by comparing a client-oriented Java CGI model with a server-oriented CGI model. In addition, a load balanced(LB, for short) CGI model is proposed, which combines two models and decides its course of action depending on the load state of the web server, and compared with the other two models. Performance evaluation models for three models are also presented. The results of computer simulations indicate that the LB CGI model performs consistently well irrespective of system load or server performance.

1. 서 론

네트워크 상에 분산된 이종 정보 자원은 보다 쉽고, 안정되고 일관된 방법으로 서비스를 제공하여야 한다. 이것은 사용자 인터페이스 측면에서는 사용하기 쉬워야하며, 사용자의 요구 사항에 적절히 대응할 수 있는 기능들을 제공하여야 한다. 또한 시스템 측면에서는 기존의 시스템들과 쉽게 통합되거나 통합할 수 있는 인터페이스를 지원해야하며, 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 기존의 이종 정보 시스템을 효과적으로 통합함으로써 구축될 수 있다.

웹은 편리한 사용자 인터페이스와 멀티미디어 환경을 제공함으로써 폭발적으로 사용자가 늘어나고 있다[1, 2]. 다양한 용도로 사용할 수 있는 웹은 모듈 인터페이스 기술을 이용하여 동적이고 다양한 서비스[3]를 가능하게 해준다. 웹과 정보 시스템 혹은 웹과 데이터베이스 시스템과의 연동[4~8]을 위한 API(application

※ 본 연구는 1998년도 한국학술진흥재단 대학부설 연구소과제의 지원에 의해 연구되었음.
† 정 회 원 : 경동정보대학 사무자동화과 교수
†† 준 회 원 : 영남대학교 대학원 컴퓨터공학과
††† 정 회 원 : 영남대학교 컴퓨터공학과 교수
논문접수 : 1999년 7월 30일, 심사완료 : 1999년 11월 6일

programming interface)나 그 시스템과의 신뢰할만한 인터페이스를 별도로 제공하지 않는 legacy 애플리케이션[9]에 대해서, 웹은 이들을 효과적으로 통합할 수 있는 모듈 인터페이스 방법을 제공한다. 이렇게 모듈 인터페이스는 다양한 장점을 갖는 웹서비스의 확장을 효과적으로 지원해준다. 이러한 이종 정보시스템과의 연동을 위한 대표적인 모듈 인터페이스에는 CGI(common gateway interface)[9]방식이 있다. 그러나 CGI 방식은 중앙 집중식 서버구조로 동시에 많은 사용자들이 서비스를 요구할 경우 웹 서버는 쉽게 과부하상태가 된다. 반면 Java를 이용한 분산구조는 웹서버가 실행 가능한 바이트코드를 클라이언트에게 전송하여 웹응용의 실행이 클라이언트에서 이루어짐으로써 웹서버의 부하가 클라이언트쪽으로 분산될 수 있다. 그러나, 이러한 사용자 요구처리의 분산은 네트워크 환경이나 클라이언트측의 컴퓨팅 능력에 따라 오히려 성능이 나빠질 수도 있다[10].

본 논문에서는 모듈 인터페이스에 의한 이종 정보시스템을 웹으로 통합할 경우, 특히 데이터베이스 시스템을 웹으로 통합하는 경우 기존 CGI를 이용한 웹서비스 구조와 Java CGI를 이용한 웹 서비스 구조, 새롭게 제안하는 load-balanced CGI를 부하분산[11~15]측면에서 분석하고 여러 가지 파라메타에 따른 세가지 방식의 성능을 시뮬레이션을 통해 평가한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 기술에 대해 설명하고, 3장에서는 성능평가를 위한 시스템 모델, 4장에서는 성능평가 결과 및 분석을 설명하고, 마지막 장에 결론이 있다.

2. 관련기술

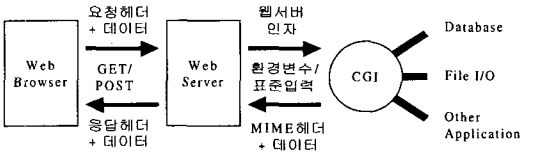
2.1 부하분산

부하분산이란 과부하 프로세싱 노드의 타스크를 저부하 프로세싱 노드로 전송하여 처리하는 것이다.[11] 이러한 부하분산의 적절한 사용으로 시스템의 처리율과 프로세싱 노드의 이용율을 증가시킨다. 전체 시스템 성능 향상이라는 측면에서 부하분산 전략에 관한 많은 연구가 있어 왔으며, 지속적인 연구가 계속되고 있다.

부하분산 전략은 크게 동적 부하분산(Dynamic Load Balancing)[12, 13]과 정적 부하분산(Static Load Balancing)[14, 15]으로 나뉜다. 동적 부하분산은 시스템의 순간적인 상태정보를 사용하므로 시스템의 현 상태에 의존적인데 반해, 정적 부하분산은 시스템의 통계적인 평균 정보를 이용하며 시스템의 순간적인 상태에는 독립적이다. 동적 부하분산은 구현에 있어서 정적 부하분산에 비해 다소 복잡하나 시스템의 순간 상태 정보들을 이용함으로써 정적 부하분산 아래서 시스템의 평균 특성에 관한 정보들을 사용할 때보다 더 나은 성능향상을 가져올 수 있다. 동적 부하분산에서 부하 이동 결정을 위한 기준으로는 대기 작업 수, CPU 이용률, 응답 시간 등이 사용된다.

2.2 CGI와 Java

CGI는 웹서버와 DB서버 등의 응용 프로그램 사이를 연결하기 위한 일종의 게이트웨이 표준이다. 이는 웹서버가 정적인 서비스를 제공하는 데 그치는 것이 아니라, 사용자의 요구에 다이나믹하게 대응할 수 있게 한다. 클라이언트가 HTTP를 통해 게이트웨이 프로그램에 접근하면 서버가 그 프로그램을 동작시키고, 클라이언트로부터 보내진 데이터를 게이트웨이 프로그램으로 전달한다. 게이트웨이 프로그램은 들어온 데이터 처리를 마치면 결과를 웹서버 측으로 보내며 서버는 이를 클라이언트 측으로 전달한다. 즉, CGI는 웹서버와 외부 또는 게이트웨이 프로그램간의 데이터들의 교환방법을 제공한다. 게이트웨이 프로그램은 C, C++, Pascal과 같은 언어로 작성되어 컴파일될 수 있거나, Perl, TCL, 혹은 다양한 쉘(shell)프로그램과 같은 스크립트 언어로 작성할 수 있다. (그림 1)은 CGI 실행 과정을 나타낸다.



(그림 1) CGI 실행 과정

웹브라우저에서는 URL(uniform resource locators)이나 폼에 의해 입력된 값을 실제 웹서버에 전달할 때 요청헤드(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

프로세서를 생성시켜 실행한다. 실행되는 프로세스는 웹 서버로부터 전달받은 인자를 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.

자바는 1991년 Sun Microsystems에서 개발이 시작되어 1993년에 웹에 적용하기로 결정하고, 1995년 SunWorld 컨퍼런스에서 공식 발표된 객체지향(object-oriented) 프로그래밍 언어와 환경이다[16]. 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[17, 18]을 도입했다.

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.

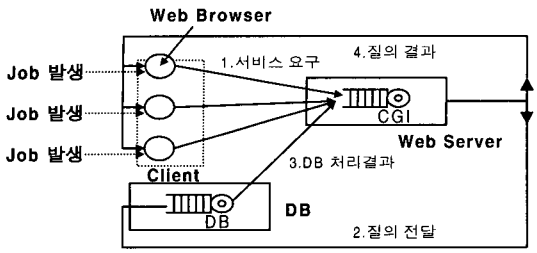
3. 부하균형 평가모델

웹 기반 정보처리 시스템의 성능에 영향을 주는 요소는 크게 사용자의 요구를 웹 서버에서 처리하는 과정, 이러한 요구를 데이터베이스에 전달하는 과정, 데이터베이스에서 처리하는 과정 등의 세 가지가 있다. 본 연구에서는 이들 요소 중 네트워크 환경은 단순화하고 웹 서버 내에서의 CGI처리, 데이터베이스에서의 처리만을 고려한 모델을 다룬다. 웹 클라이언트 프로세서들은 웹 서버와 동일한 LAN 환경에서 동작하게 함으로써, 네트워크 환경으로 인한 병목의 영향을 최소화하여 성능 평가의 결과에 대한 신뢰성을 보장한다.

일반적으로 사용하는 CGI 방식은 전형적인 중앙 집중식 서버구조[4]로서 하나의 서버가많은 응답을 처리해야 한다. 사용자가 질의를 웹서버에 보내면 서버는 입력된 데이터를 분석해서 에러가 있다면 웹서버를 거쳐 웹브라우저에게 에러 메시지를 보내고, 에러가 없다면 실행을 해서 결과를 웹서버를 거쳐 웹브라우저에게 보내게 된다. 이러한 구조의 문제점은 간단한 에러처리조차 웹서버를 거쳐서 처리해야 하고, 동시에 많은 클라이언트의 서비스 요구시 서버가 과부하상태가 됨으로써 서버에서 병목 현상을 일으킬 수 있다. 이러한 CGI 프로그램을 Java로 구현함으로써 서버측에서 처리해야할 부하를 클라이언트측으로 이동시킬 수 있다.[10] 본 연구에서는 기존의 서버 지향 CGI 모델(이후 CGI 모델)과 클라이언트 지향 Java CGI모델(이후 Java CGI 모델), 그리고 서비스 요구가 도착하면 서버의 부하상태에 따라 서버 지향 CGI 모델과 클라이언트 지향 Java CGI를 적당히 선택하여 서비스를 처리하는 부하균형 CGI 방식(이후 LB CGI 모델)의 평가모델을 제시하고 성능을 서로 비교 분석해본다.

3.1 서버 지향 CGI 방식의 성능평가 모델

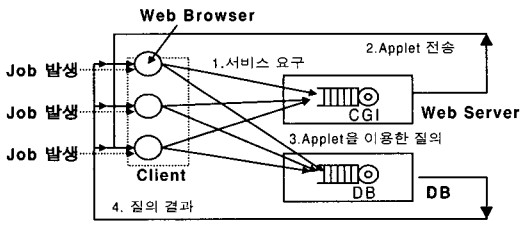
CGI 실행모듈 방식은 웹과 네트워크 정보시스템 혹은 데이터베이스 시스템을 통합시키는 가장 단순한 방법으로, 기존의 웹서버, 웹 브라우저, 그리고 웹 관련기술(예, URL, HTTP, HTML 등)을 변경없이 사용하여 다양한 정보시스템을 웹 기반의 인트라넷 정보시스템화 할 수 있다. 하지만 동시에 많은 처리가 요구될 경우, 각 요구에 대한 CGI 프로세서가 매 번 생성되고 그 때마다 데이터베이스 연결, 파일 I/O 등이 수행된다. 또한 프로세서의 생성 및 종료, 프로세서들간의 통신 및 자료복사, 그리고 빈번한 프로세서 교체 등의 원인으로 웹서버 시스템의 자원 부족 현상이 쉽게 발생하여 전체 시스템의 성능을 저하시킨다. (그림 2)는 서버 지향 CGI 실행 모듈 방식의 평가모델을 보여 주고 있다.



(그림 2) CGI 실행모듈 방식의 성능 평가 모델

3.2 클라이언트 지향 Java CGI 방식의 성능평가 모델

플랫폼에 독립적인 특징으로 인해 분산형 시스템을 쉽게 구축할 수 있는 자바 방식은 웹클라이언트와 데이터베이스간의 효율적인 분산형 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 사용자 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. (그림 3)은 클라이언트 지향 Java CGI 방식의 평가 모델이다.



(그림 3) 자바 방식의 성능 평가 모델

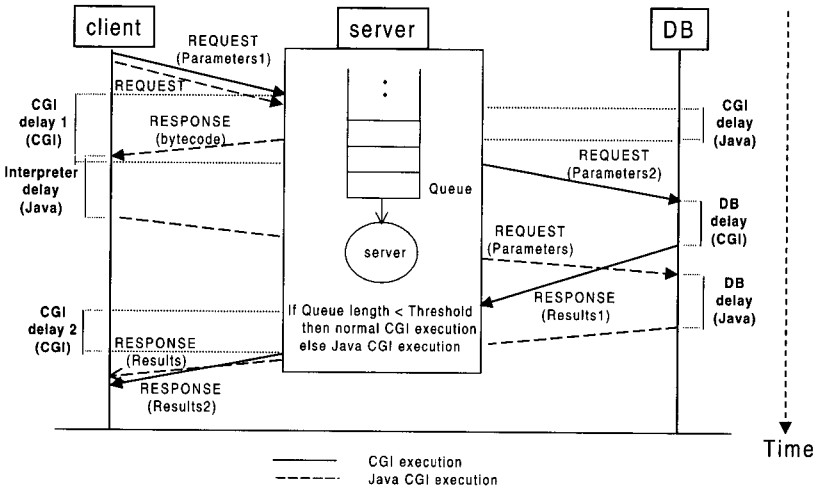
3.3 CGI 모델과 Java CGI 모델의 처리과정 비교

사용자가 웹브라우저를 통해 서비스 요구시 서버지향 CGI 모델과 클라이언트 지향 자바 모델은 4단계의 과정을 거친다. CGI 모델의 경우 웹클라이언트의 요구는 웹서버를 거쳐 데이터베이스에게 전달되고, 처리결과는 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 자연스럽게 웹서버의 부하를 웹클라이언트에게로 분산함으로써 CGI 모델에 비해 부하분산의 효과를 가질 수 있다. 그러나 부하가 저부하일때는 일반적으로 서버지향 CGI 방식의 성능이 우수하며, 과부하일때는 클라이언트 지향의 Java CGI 방식의 성능이 우수하다.따라서 각 방식의 특징을 살릴 수 있는 부하균형 방식이 필요하다.

3.4 부하균형 CGI 방식

서버지향 CGI 모델과 클라이언트 지향 Java 모델을 결합하고 웹 서버의 부하상태에 따라 두 방식중 한 방식으로 서비스방식을 결정하여 요구를 처리하는 부하균형 CGI 모델(LB CGI 모델)을 제안한다. 이 모델에서는, 클라이언트가 서비스를 요청하면 서버가 과부하 상태인지 아닌지를 먼저 점검한다. 만약 서버가 과부하상태이면 그 서비스는 Java CGI 방식으로 처리되고, 과부하 상태가 아니면 일반적인 CGI 방식으로 처리된다. 이러한 LB CGI방식을 사용할 경우, 클라이언트들의 요구 부하에 따라 요구를 적절히 웹서버와 클라이언트간에 분산시키므로 시스템 전체의 성능을 항상 최적의 상태로 유지시킬 수 있다. (그림 4)는 LB CGI 모델의 처리과정을 보여준다.

본 연구에서 제안하는 LB CGI 모델은 큐길이를 기반으로 임계값(Threshold) 정책을 사용하는 동적부하분산 전략[11]을 이용한다. 임계값 정책이란 한 타스크가 프로세싱 노드에 들어올 때 그 노드의 큐 길이(서비스를 기다리거나 처리중인 타스크의 수)가 임계값보다 크거나 같으면 그 노드는 다른 노드로 타스크 처리



(그림 4) LB CGI 방식의 처리과정

를 넘긴다. 본 논문에서의 LB CGI 모델은, 클라이언트가 웹 서버로 서비스를 요청할때마다 웹 서버의 큐 길이를 조사하여 임계값보다 크거나 같으면 서버가 과부하상태인 것으로 판단하여 Java CGI 방식으로 서비스를 처리하고 큐 길이가 임계값보다 작으면 일반적인 CGI 방식으로 서비스를 처리한다. 본 시뮬레이션에 사용된 임계값은 1로 두었다. 일반적으로 부하분산에서 임계값 정책을 사용할 경우 임계값은 1 또는 2 가 적절한 것으로 알려져 있다[13]. 본 시뮬레이션에서도 임계값을 다양하게 변화시켜 본 결과 임계값이 1 일때가 부하분산의 효과가 큰 것으로 나타났다.

성능 평가 모델의 파라메터로는 사용자의 요구를 처리하기 위해 웹서버에서 수행하는 CGI 프로세서의 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 모델과 Java CGI 모델의 시뮬레이션 파라메타가 <표 1>에 나타나 있다. 웹 연동 정보시스템에서 트랜잭션의 실행을 실제 측정한 결과가 나타난 논문을 참고하여 파라메타를 정하였다[16, 17, 19, 20].

CGI 모델에서 첫 번째 CGI 지연은 CGI 프로세서 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. Java CGI 모델에서의 CGI 지연은 CGI 프로세서 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. <표 1>의 파라메타를 평균값으로 가지는 지수분포로 각 지연시간이 결정된다. LB CGI 모델은 각 클라이언트의 서비스 요청을 CGI 모델 또는 Java CGI 모델 두가지 방식중 하나로 수행하므로 위의 파라메타가 그대로 적용된다. 웹 서버가 과부하인지를 판단하는 기준으로는 웹 서버의 처리를 기다리는 클라이언트 서비스의 요청 갯수가 사용되었다.

〈표 1〉 시뮬레이션 파라메터

(단위 msec)

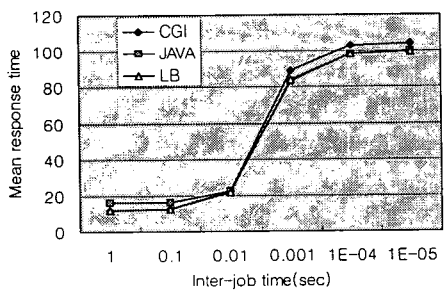
	Java interpreter delay	CGI delay	DB delay
CGI 모델		0.3, 0.2	0.5
Java CGI 모델	0.6	0.3	0.5

시뮬레이션 모델은 동시 접속 상황이 고려대상이므로 동시 접속자를 20명으로 가정하여 20개의 클라이언

트에서 사용자 요구를 발생시키고, 각 클라이언트에서는 발생된 작업에 대한 응답이 돌아오면 작업 전송 간격에 따라 다음 작업을 계속 발생시킨다. 그리고 성능 평가를 위한 시뮬레이터로는 GPSS/H를 이용하였다.

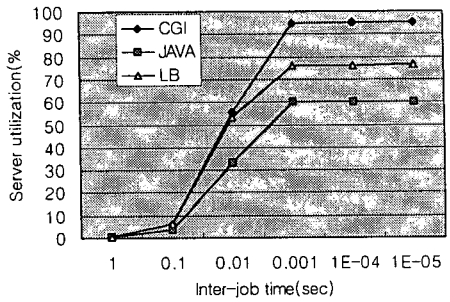
4. 성능평가 결과 및 분석

CGI, Java CGI 그리고 본 연구에서 제안하는 LB CGI 방식의 성능 평가 모델을 기반으로 작업 전송 간격에 따른 평균 응답시간, 데이터베이스 이용률, 그리고 서버 이용률을 측정하였다. 여기서 작업 전송 간격이란 클라이언트인 브라우저에서 하나의 작업을 요구하고 그 작업에 대한 응답을 받은 후 다음 작업을 요구할 때까지의 간격이며, 간격이 작을수록 웹서버의 부하는 높은 것으로 본다. 데이터베이스 이용률이란 데이터베이스가 이용되는 시간/전체 시뮬레이션 시간을 나타내고, 서버 이용률이란 서버가 이용되는 시간/전체 시뮬레이션 시간을 나타낸다.



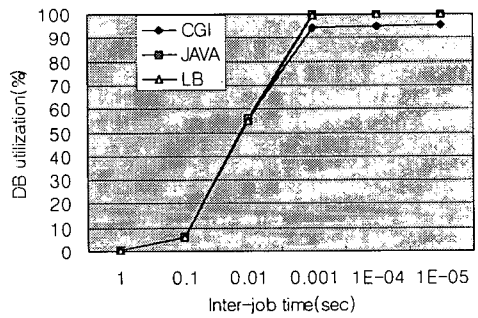
(그림 5) 클라이언트 작업 전송 간격에 따른 평균 응답시간

(그림 5)는 클라이언트의 작업 전송 간격에 따른 CGI 모델, Java CGI 모델과 LB CGI 모델의 평균 응답시간을 보여 주고 있다. 웹 서버가 저부하일 때는 CGI 모델이 Java CGI 모델에 비해 우수하다. 그러나 부하가 증가함에 따라 Java CGI 모델이 CGI 모델에 비해 평균응답시간이 우수함을 보인다. 서버가 과부하일때는 서비스를 클라이언트 측에서 처리하는 것이 더 효율적임을 나타낸다. LB CGI 모델은 서버의 부하상태에 따라 두 방식 중 한 방식으로 처리하므로 웹 서버가 저부하일 때 CGI 모델과 비슷한 성능을, 웹 서버가 과부하일때는 Java CGI 모델과 유사한 성능을 보인다.



(그림 6) 작업 전송 간격에 따른 웹 서버 이용률

(그림 6)은 작업 전송 간격에 따른 CGI 모델, Java CGI 모델과 LB CGI 모델의 웹 서버 이용률을 보여주고 있다. 웹서버의 부하가 점점 증가함에 따라 CGI 모델이 Java CGI 모델에 비해 서버 이용률이 급격하게 증가함을 보인다[19]. 이에 비해 Java CGI방식은 서버가 할 일을 클라이언트 측에서 처리하므로 그 만큼 서버 이용률이 낮다. LB CGI 모델은 서버의 부하상태에 따라 두 방식 중 한 방식으로 처리하므로 서비스를 완전히 클라이언트 측에서 처리하는 Java CGI 방식보다는 높고, 완전히 서버 측에서 처리하는 CGI 방식보다는 낮은 중간 값을 가지는 것을 볼 수 있다.



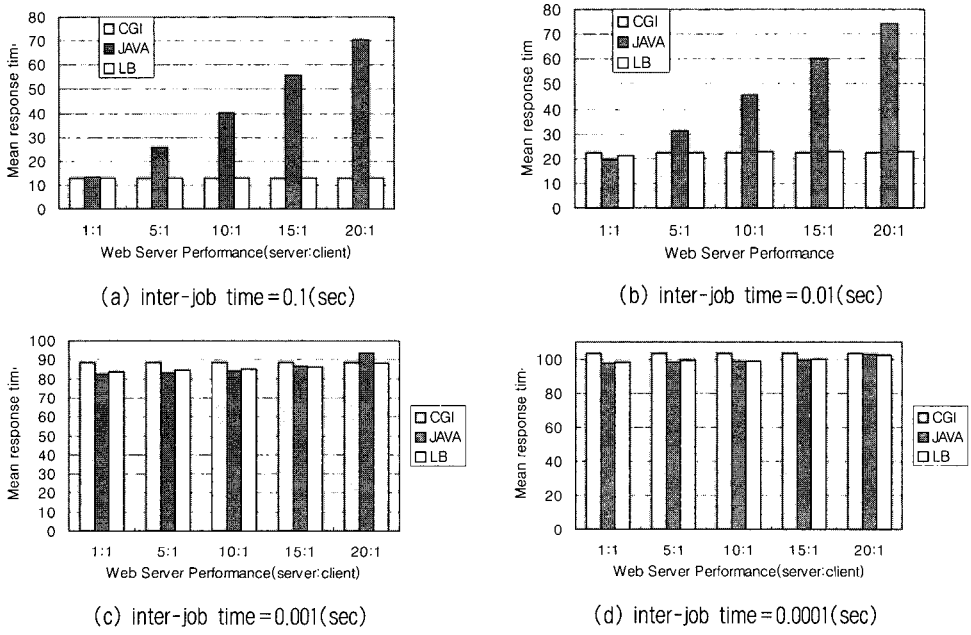
(그림 7) 작업 전송 간격에 따른 데이터베이스 이용률

(그림 7)은 작업 전송 간격에 따른 CGI 모델, Java CGI 모델과 LB CGI 모델의 데이타베이스 이용률을 보여 주고 있다. 웹 서버가 저부하일 때, 세 모델의 이용률 차이는 거의 나타나지 않지만, 부하가 점점 증가함에 따라 Java CGI 모델과 LB CGI 모델의 데이타베이스 이용률이 높아진다. CGI 모델은 부하가 서버에 집중될 때, 서버의 과부하로 인해 CGI가 처리하는 프로세스 수가 적어지므로 당연히 DB 이용률도 줄어든

다. 하지만 Java CGI 나 LB CGI 방식은 서버 부하시 CGI 가 할 일을 클라이언트에게 넘겨줌으로 클라이언트에서 DB 연결을 함으로 DB 이용률이 CGI 방식보다 높다.

웹 서버의 성능이 각 시스템에 미치는 영향도 살펴보았다.

(그림 8)은 서버의 성능 변화에 따른 세 모델의 평균 응답시간을 보여주고 있다. 서버의 성능변화를 위해, 서버의 처리시간(CGI delay)이 아닌 클라이언트 처리시간(Java interpreter delay)을 변화시켜 시뮬레이션 하였다. (그림 8)의 가로축에서 예를 들어 5 : 1이 뜻하는 것은 클라이언트의 성능이 서버 성능의 $\frac{1}{5}$임을 나타낸다. (그림 8)의 (a)는 비교적 시스템이 저부하인 상태를 나타내고 (b), (c), (d)로 갈수록 과부하가 되는 상태를 나타낸다. (그림 8)의 (a)에서, 서버와 클라이언트의 성능이 같을 때는 세 모델이 모두 비슷한 평균 응답시간을 나타내나 서버 성능이 증가할수록 Java CGI 모델의 응답시간이 급격하게 증가함을 볼 수 있다. 이것은 서버의 성능이 우수할 때는 서비스를 클라이언트 측에서 처리하지 않고 서버 측에서 처리하는 것이 효율적이기 때문이다. (그림 8)에서 inter-job time이 작을수록 시스템이 과부하상태이므로 갈수록 세 모델의 응답시간은 증가하는 추세이다. 특히 시스템이 과부하 상태로 갈수록 CGI 모델의 응답시간이 급격히 증가하는 것을 볼 수 있다. 시스템이 과부하일수록, 서비스를 서버 측에서 프로세서를 생성시켜 처리하는 CGI 방식은 서버의 부하가 증가되어 그만큼 응답시간이 느려지는 것이다. (그림 8(d))에서, 시스템이 과부하 상태일 때, 서버와 클라이언트의 성능이 같을 때는 Java CGI 방식이 조금이나마 성능이 좋으나 서버의 성능이 우수할수록 별 차이가 없음을 볼 수 있다. LB CGI 방식은 시스템의 부하상태나 서버의 성능에 상관없이 항상 좋은 성능을 보이는 모델과 유사한 결과를 가지는 것을 (그림 8)에서 알 수 있다. 이것은 LB CGI 모델이 서버의 부하상태에 의하여 과부하 상태일 때는 서비스를 클라이언트 측에서 분산 처리하는 Java CGI 방식으로 서비스를 처리하고 저부하일 때는 중앙 집중식 서버구조인 CGI 방식으로 서비스를 처리하기 때문이다. (그림 8)에서 보이듯이 전체적으로 볼 때 Java CGI 모델은 시스템이 저부하 상태이고 서버의 성능이 우수할수록 성능이 떨어지고 CGI 모델은 시스템이 과부하 상태일 때는 성능이 떨어진다. 그러나 LB CGI 모델을 사용하면 서버의 성능이나 시스템의 부하 상태에 상관없이 항상 최상의 성능을 가질 수 있다는 것을 보이고 있다.



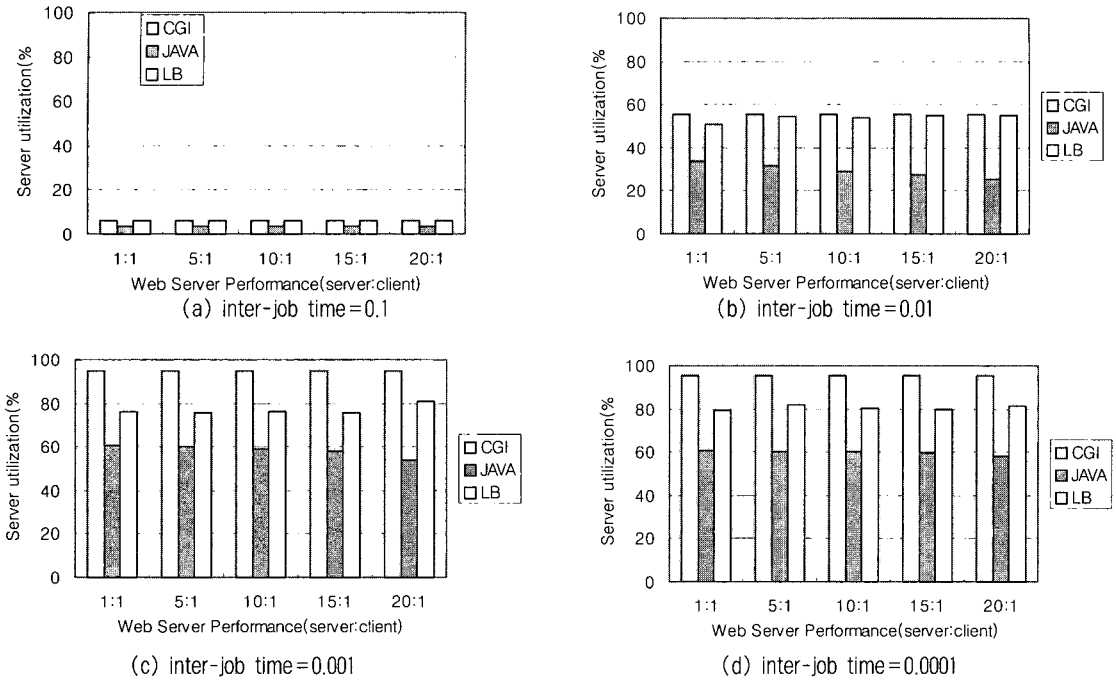
(a) inter-job time = 0.1(sec)

(b) inter-job time = 0.01(sec)

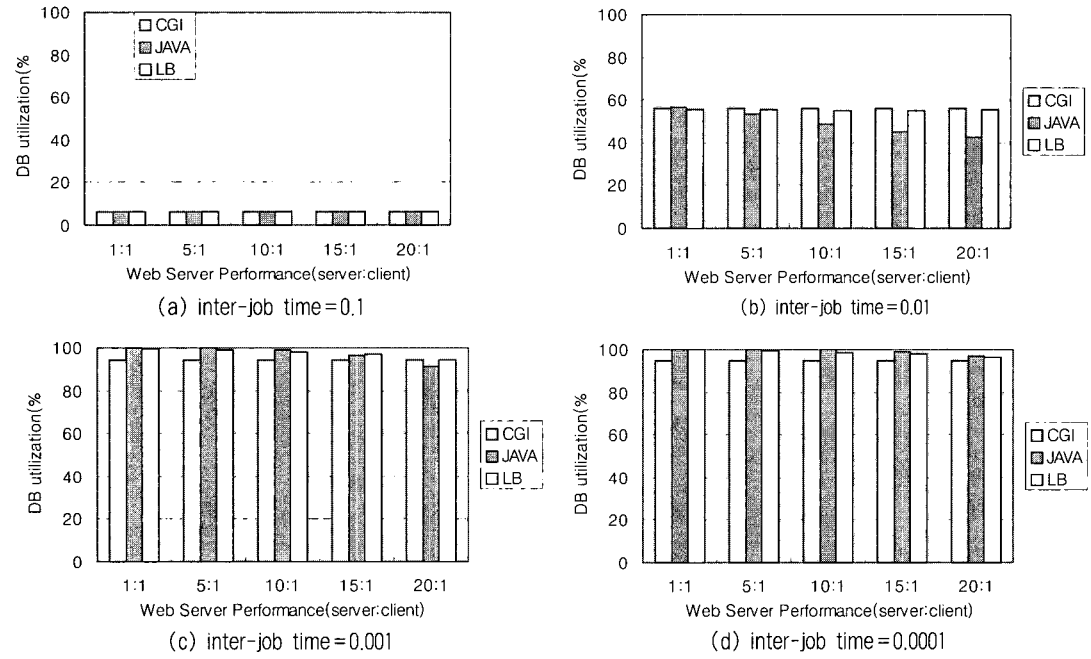
(c) inter-job time = 0.001(sec)

(d) inter-job time = 0.0001(sec)

(그림 8) 서버 성능변화에 따른 평균 응답시간



(그림 9) 서버 성능변화에 따른 서버 이용률



(그림 10) 서버 성능변화에 따른 데이터베이스 이용률

(그림 9)는 웹 서버의 성능변화에 따른 세 모델의 서버 이용률을 보여주고 있다. (a)는 시스템 전체에 부하가 적을 때로, 서버이용률은 서버가 이용되는 시간/전체 시뮬레이션 시간을 나타내므로 전체적으로 이용률이 낮게 나타난다. (그림 9)에서, Java CGI 방식은 서비스를 클라이언트 측에서 처리함으로 다른 두 모델에 비해 서버 이용률이 낮게 나타난다.

(그림 10)은 웹 서버의 성능 변화에 따른 세 모델의 데이터베이스 이용률을 보여주고 있다.

(a)는 시스템 전체에 부하가 작을 때로, 데이터 베이스 이용률은 데이터베이스가 이용되는 시간/전체 시뮬레이션 시간을 의미하므로 서버이용률과 마찬가지로 전체적으로 이용률이 낮게 나타난다. (b)에서, 서버와 클라이언트의 성능이 같을 때는 데이터베이스 이용률이 세 모델 모두 비슷하나 서버의 성능이 우수할수록 Java CGI 방식의 이용률이 낮아짐을 볼수 있다. 이것은 Java CGI 방식은 서비스를 클라이언트 측에서 처리함으로, 서버의 성능이 우수할 때는 서비스를 서버 측에서 처리하는 CGI 방식보다는 프로세스 처리수가 그만큼 적다는 것을 의미한다. 그에 비해 LB CGI 방식은 서버의 부하상태에 따라 두 방식 중 한 방식으로 처리하므로 서버의 성능이 우수할 때는 CGI 방식으로 서비스를 주로 처리하기 때문에 CGI 방식과 비슷한 데이터베이스 이용률을 가진다.

이와 같은 성능 평가 결과에서 웹 서버가 고부하 상태로 갈수록, 많은 웹 자원을 사용하는 CGI 모델은 단위시간당 웹서버가 처리할 수 있는 사용자 요구 수와 이에 따른 데이터베이스와의 연결 수를 제한하였다. 하지만 Java CGI 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.

한편 LB CGI 모델은 서버 이용률이 CGI 모델보다 낮고 Java CGI 모델보다는 높은 중간 정도의 이용률을 보인다.

웹 서버와 클라이언트의 성능이 같고 시스템이 저부하 상태일 때는 세 모델 모두 거의 비슷한 성능을 가지지만, 웹 서버와 클라이언트의 성능이 같고 시스템이 과부하 상태일 때는 웹 서버의 부하 증가로 인하여 CGI 모델이 다른 두 모델에 비해 성능이 떨어진다. Java CGI 모델에서는 웹 서버의 부하가 클라이언트 쪽으로 분산되므로, 웹 서버의 성능이 클라이언트에 비해 상대적으로 클수록 Java CGI 모델의 성능은 급격히 떨어진다. 그러나 시스템이 과부하상태일 때는 서버의 성능이 클라이언트에 비해 증가하더라도 CGI 모델의 성능이 가장 떨어진다. LB CGI 모델은 웹 서버의 상태에 따라 서비스 방식을 결정하므로 시스템 부하나 서버의 성능에 상관없이 비교적 최상의 성능을 가져온다.

5. 결 론

본 연구에서는 이종 정보시스템의 웹 통합방식인 중앙 집중식 서버구조의 CGI 모델과 클라이언트 분산구조의 Java CGI 모델을 기본으로 하고 서버의 상태에 따라 요구된 작업에 대해 두 방식 중 하나의 서비스 방식을 결정하는 새로운 LB CGI 모델을 제안하고 두 모델과 비교 분석해 보았다. CGI 모델은 비교적 구현이 간단하나 동시에 많은 서비스 요구로 인한 웹 서버 자원의 병목현상으로 전체 웹 서버 성능 저하를 가져온다. Java CGI 모델은 처음 로딩시 인터프리터되는 시간이 많이 소요되지만 동시에 많은 사용자가 서비스를 요구할 경우 서비스 모듈 자체가 클라이언트에서 실행되므로 CGI 모델에 비해 웹 서버의 부하를 경감시킬 가능성이 높다. 그러나 웹 서버의 성능이 클라이언트에 비해 탁월하고 시스템이 저부하 상태일 때는 Java CGI 모델이 CGI 모델이나 LB CGI 모델에 비해 성능이 떨어지는 단점이 있다. 또한 LB CGI 모델은 서버의 상태에 따라 서비스 방식을 결정하므로 서버의 성능이나 시스템의 부하상태에 상관없이 CGI 모델이나 Java CGI모델과 비교하여 상대적으로 좋은 성능을 가짐을 알 수 있었다.

현재 이종 정보시스템과의 웹 통합은 주로 구축의 편이성으로 인해 중앙 집중식 서버구조인 CGI 방식을 사용하고 있다. 하지만 인터넷을 통한 사용자 요구의 다양화와 데이터 양의 급격한 증가로 인해 웹 서버의 부하는 점점 증가하고 있다. 그러나 서버의 성능 역시 급속히 향상될 가능성이 있으므로 본 논문에서 제안한 LB CGI 모델과 같은 하이브리드 형식을 기반으로 이종의 다양한 정보시스템을 웹과 통합함으로써 효과적인 정보시스템을 구축할 수 있을 것이다.

참 고 문 헌

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

[2] 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.

[3] Yongjun Choi, Kyungsu Lim, Dosam Hwang, Chonggun Kim, "A Management Method for Hierarchical Information Structures on Web Systems," The transactions of the Korea information processing society, Vol.5, No.5, pp.1300-1310, 1998.

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

[5] Nick N.Duan, "Distributed Database Access in a Corporate Environment Using JAVA," *The Fifth International World Wide Web Conference*, 1996.

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

[7] 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.

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

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

[10] 임인택, 백승구, 임경수, 김수정, 김종근, "Java CGI에 의한 이종정보시스템의 웹통합 연구", 한국통신학회논문지, 제24권, 제3B호, pp.399-409, 1999.

[11] D. Eager, E. Lazowska, and J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed System," IEEE Trans.SE., Vol.SE-12, No.5, pp.662-675, 1986.

[12] D. Eager, E. Lazowska, and J. Zahorjan, "Comparison of Receiver-initiated and Sender-initiated Adaptive Load Sharing," Performance Evaluation 6, pp.53-68, 1986.

[13] 임경수, 김수정, 김종근, "스타형 컴퓨터 네트워크의 부하균형방향 정책", 정보처리 응용학회 논문지, 제1권, 제4호, pp.427-437, 1994.

[14] A. N. Tantawi and D. Towsley, "A General Model for Optimal Static Load Balancing in Star Network Configurations," In E.Gelembe, editor, PERFORMANCE'84, pp.277-291, Elsevier Science Publisher B.V.(North-Holland), 1984.

[15] C. Kim and H. Kameda, "An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems," IEEE Trans.Comp., Vol.41, No.3, pp.381-384, March 1992.

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

[17] 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.

[18] Drew Dean, Edward W.Felten, Dan S. Wallach, "JAVA Security:From HotJAVA to Netscape and Beyond*," *IEEE Symposium on Security and Privacy*, 1996.

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

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

김 수 정

e-mail : sjkim@kyungdong-c.ac.kr

1991년 영남대학교 공과대학 전산기공학과 학사

1994년 영남대학교 대학원 전산기공학과 정보처리전공(석사)

1996년 영남대학교 대학원 전산기공학과 컴퓨터 시스템전공 박사 수료

1997년~현재 경동정보대학 사무자동화과 전임강사

관심분야 : 분산처리, 성능평가, 인터넷, 정보통신

백 승 구

e-mail : sgback@nety.yeungnam.ac.kr

1997년 경일대학교 제어계측공학과 졸업(공학사)

1999년 영남대학교 컴퓨터공학과 석사과정 수료

관심분야 : Network protocol analysis, network design and analysis, Java system

김 종 근

e-mail : cgkim@ynucc.yeungnam.ac.kr

1981년 영남대학교 공과대학 전자공학과 학사

1987년 영남대학교 대학원 전자공학과 계산기전공 석사

1991년 (일본)전기통신대학 정보공학과 박사

1984년~1990년 경북전문대학 전산과 전임강사

1996년~1997년 (미국) Virginia Tech. 연구교수

1991년~현재 영남대학교 컴퓨터공학과 부교수

관심분야 : 분산시스템, 컴퓨터통신 기술, 차세대 인터넷 기술, 성능평가기술 임