대한기계학회 2001년도 춘계학술대회논문집 E pp. 488~493 KSME 01S596

인트라넷 안에서 효율적인 수치해석을 구현하기 위한 자바기술

송희용*·고성호**

Java Technology for Implementing Efficient Numerical Analysis in Intranet

Heeyong Song, Sungho Ko

Key Words: Numerical Analysis(수치해석), Java(자바), Applet(애플릿), Servlet(서블릿), RMI(원격 함수호출), Thread(쓰레드)

Abstract

This paper introduces some useful Java technologies for utilizing the Internet in numerical analysis, and suggests one architecture performing efficient numerical analysis in the Intranet by using them. The present work has verified it's possibility by implementing some parts of this architecture with two easy examples. One is based on Servlet-Applet communication, JDBC and Swing. The other is adding multi-threads, file transfer and Java Remote Method Invocation to the former. Through this work it has been intended to make the base for the later advanced and practical research that will include efficiency estimates of this architecture and deal with advanced load balancing.

기호설명

ρ	: 밀도
P	: 압력
$\vec{V}$	: 속도벡터
t	: 시간
μ	: 점성계수
T	: 온도
α	: 열 확산계수

1. 서 론

인터넷은 그 관련 기술들이 빠르게 발전하면서 다른 많은 산업을 흡수 또는 지원하고 있다. 이는 인터넷을 활용하여 많은 이점을 얻을 수 있기 때문이며 어떤 분야도 예외일 수는 없다. 인터넷의 가장 큰 장점은 거리에 구애받지 않고 필요한 자원에 접근 또는 공유할 수 있다는 것이다. 이런 인터넷의 장점을 공학의 한 분야인 수치해석에 활용할 수 있는가를 확인하는 것이 본 연구의 목적이다.

그럼에도 불구하고 이 논문의 범위를 인트라넷에 국한한 이유는 단지 데이터의 전송속도와 보안에 대한 질의를 피하기 위함이지 본 연구에서 제안된 구성을 인터넷으로 확장할 수 없음을 의미하는 것이 아니다. 또, 실제로 인터넷보다는 인트라넷에서 더 효율적이기 때문이기도 하다. 따라서, 본 논문에서는 인트라넷과 인터넷을 같은 개념으로 다룬다.

자바는 인터넷이 활성화된 이후에 만들어진 언어로서 인터넷 기반의 제품 개발에 필수적인 수단으로 떠올랐다. 자바의 가장 큰 장점은 시스템에 대한 독립성과 이동성이다. 자바 객체는 시스템의 종류에 상관없이 인터넷에 연결되어 있고 자신을 활성화시켜줄 가상기계가 존재한다면 언제든지 이동하여 실행될 수 있다. 이런 특성

* 충남대학교 대학원
** 충남대학교 기계설계공학과

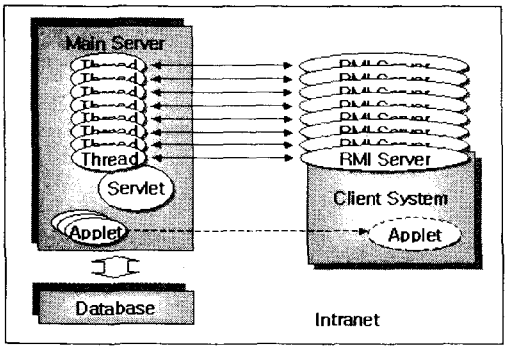

Fig. 1 Architecture performing efficient numerical analysis by using Java technology

을 잘 활용한다면 인터넷 안에서 보다 효율적인 수치해석을 수행할 수 있을 것이다. 또 다른 자바의 장점은 재사용성이다. 이는 자바와 같은 객체지향 언어의 일반적인 특성으로서 전문 프로그래머가 아니더라도 손쉽게 자신의 목적에 맞는 프로그램을 개발하고 유지 및 보수할 수 있도록 해 주며 이는 자바의 확장성으로도 설명할 수 있다.

현재 자바의 장점을 수치해석에 활용하고자 하는 연구는 개척단계에 있으며 주로 추상적인 구성의 제시나 분산프로그래밍에 국한되어있다. 이에 본 연구에서는 자바의 장점을 실제에 적용하고자 웹 애플리케이션을 기반으로 하여 모든 GUI(Graphic User Interface)를 인터넷 브라우저 내에서 확인할 수 있도록 구현하는데 초점을 맞추었다.

Fig. 1은 자바의 특성을 활용하여 구현이 가능하리라 생각되는 해석 과정을 보여주고 있다. 인트라넷 안에 있는 컴퓨터들은 서버를 중심으로 계산에 참여할 수도 있고 계산을 요청할 수 있도록 구성된다. 전처리 과정과 후처리 과정은 주 서버에서 제공되는 애플릿을 통해 사용자의 시스템에서 이루어진다. 전처리 과정을 마친 애플릿에서 요청된 계산은 다시 서버에 의해 인트라넷 내의 모든 컴퓨터로 분산되어 수행되며 그 과정 동안 사용자에게 애플릿이 전달되고 애플릿-서블릿 교신(Applet-Servlet Communication)[1]을 통해 사용자에게 수행 과정에 대한 정보를 주게 된다.

물론 이 구성 전체를 구현하는 일은 거대한 작업이 될 것이다. 따라서 본 연구에서는 이를 인터넷을 통한 전처리 및 후처리 과정의 가능성을 확인하는 부분과 웹 애플리케이션을 기반으로 한 분산 수치해석을 구현하는 부분으로 나누어 간단한 예제를 구현해봄으로써 제시된 구성의 가능성을 확인하고 추후의 효율성 평가를 포함하게 될 실질적이고 진보적인 연구를 위한 기반을 마련하고자 하였다.

2. 웹 애플리케이션으로 구현한 수치해석

2.1 급 확대 유동 해석

많은 물리적인 현상들은 적당한 가정 안에서 미분방정식으로 표현되며 이를 지배방정식이라 한다. 그러나, 아주 단순한 경우를 제외한 대부분의 문제들은 해석적으로 해결할 수 없어 이산화 과정을 거쳐 컴퓨터의 도움을 받게 된다. 이렇게 계산된 수치해석의 결과는 미리 근사적인 경향을 제공함으로써 많은 기간과 비용이 소요되는 실험의 단점을 보완하고 있다. 구현된 예제는 급격히 확대되는 파이프에서의 유체유동을 해석하고 있다[2].

유체유동의 지배방정식은 연속방정식과 Navier-Stokes 방정식이 있으며 다음과 같다.

$$\frac{\partial}{\partial t}(\rho)+\nabla\cdot(\rho\vec{V})=0 \qquad (1)$$

$$\frac{\partial}{\partial t}(\rho\vec{V})+\nabla\cdot(\rho\vec{V}\vec{V})=\nabla\cdot(\mu\ \nabla\vec{V})-\nabla P \qquad (2)$$

식 (2)에서 각 항들은 순서대로 비정상항(unsteady term), 대류항(convection term), 확산항(diffusion term), 생성항(source term)을 나타내고 ρ, μ는 각각 밀도와 점성계수를 의미한다.

위 식을 난류 유동에 적용하기 위하여 시간으로 평균하면 RANS(Reynolds averaged Navier-Stokes) 방정식이 얻어진다. 이때 우변의 점성항에는 $-\rho\overline{u'_i u'_j}$ 항이 추가되며 이를 레이놀즈응력(Reynolds stress)항이라고 한다. 이 레이놀즈 응력항은 Boussinesq의 와점성(eddy viscosity) 개념과 $k-\varepsilon$ 모델을 사용하여 구한다.

위의 지배방정식은 Patankar의 SIMPLE (Semi-Implicit Method for Pressure-Linked Equation) 알고리즘(algorithm)[3]을 사용하여 계산된다.

입구의 경계조건은 유동의 속도를 일정 값으로 하고, 난류에너지는 주류 운동에너지의 3%로 하며 소산율은 난류의 길이 스케일을 확대유로 폭

의 1%로써 산출하고 있다. 출구의 경계조건으로는 각 값의 유동방향 구배를 0으로 준다.

유동방향을 따라 등비급수적으로 배치한 16 × 16의 격자를 사용하고 있으며 계산영역은 상하대칭을 가정하여 위쪽 반만을 대상으로 한다.

2.2 웹 애플리케이션 구현

프로그램 소스를 전환하는 일은 일반적인 생각과 달리 매우 단순한 작업이다. 여기에서는 포트란으로 작성된 소스를 자바로 전환하였으며 이를 애플릿(Applet)과 서블릿(Servlet)으로 분리하였다.

애플릿은 자바를 세상에 알리는 매우 중요한 역할을 담당했다. 웹 상에서 애플릿이 제공했던 동적인 모습들은 사람들의 주목을 끌기에 충분했다. 그러나, 이제 애플릿 만으로는 자바의 강력해진 기능들을 충분히 활용할 수 없게 되었다. 애플릿은 웹서버에 의해 자신이 포함된 문서가 요청될 때 함께 다운되어 JVM(Java Virtual Machine)에 의해 실행된다. 결국, 브라우저를 통해 접근한다는 점 이외에 일반 애플리케이션과 전혀 다른 점이 없는 것이다. 따라서, 애플릿은 서버와 교신하지 않으면 사용자들이 원하는 동적인 내용을 전달할 수 없다. 애플릿은 서버와 교신할 때 비로소 웹 애플리케이션의 일원으로서 임무를 수행할 수 있게 된다. 애플릿의 GUI는 JFC(Java Foundation Class)를 사용하여 구현하였다. JFC는 기존의 AWT(Abstract Window Toolkit)에 비해 훨씬 다양하고 화려한 GUI를 구현할 수 있게 하며 스윙(Swing)[4]이라고도 불린다.

웹 애플리케이션은 일반적으로 3-차원(tier) 모델을 따른다. 사용자에게서 데이터를 모아 웹서버에 요청을 보내고, 웹서버는 요청된 서버 프로그램을 실행시켜 사용자에게 보여줄 수 있도록 데이터를 묶어 브라우저로 다시 보낸다. 자바에서는 이를 구현하기 위해 서버 측 방안으로 확장성과 이식성이 뛰어난 서블릿[5]이라는 기술을 제공한다. 자바 서블릿은 기존 개발자들에 의해 많이 사용되고 있는 CGI(Common Gateway Interface)에 비해 매우 효율적이다. CGI가 요청에 대해 프로세스(process)를 생성하는 반면 서블릿은 경량의 쓰레드로 처리하여 시스템의 부담을 줄일 수 있고 또 모든 Java APIs(Application Programming Interface)에 접근할 수 있어 확장이 용이하다. 애플릿이 서버와 통신하는 것은 매우 어려운 일이었다. 그러나, 서블릿은 서버에 접속하기 쉬운 자바 기반의 중간 단계 역할을 수행함으로써 이러한 작업을 손쉽게 만들어 준다.

또, 추후 데이터베이스 사용 가능성을 함께 검증하기 위해 공통의 저장소로서 굳이 데이터베이스를 사용하고 있다. 오늘날에는 많은 제품들이 프로세스 중심이 아닌 데이터 중심으로 개발되고 있다. 따라서 개발 도구는 데이터베이스 접근을 위한 견고하고 간단한 수단을 제공해야 한다. 그러나 대부분의 데이터베이스 시스템은 그것을 제어하기 위해 자사 고유의 수단을 제공하고 있다. 따라서 개발자들은 실행 환경에 맞추어 코드를 작성해야 할뿐만 아니라, 접속하고자 하는 데이터베이스의 종류에 따라 서로 다른 함수들을 사용해야 한다. 이를 해결하기 위해 자바는 JDBC(Java Database Connectivity)[6]를 제공한다. JDBC는 제공업체 고유의 인터페이스를 이용하여 데이터베이스 연동을 수행하며 이들 인터페이스들은 각각 서로 다른 개별적인 제공업체들에 의해 구현되었다. 특정한 데이터베이스 엔진을 위해 JDBC 인터페이스를 구현하는 클래스들을 JDBC 드라이버라고 한다. JDBC는 각 업체 고유의 내용들을 추상화하고, 일반적인 함수들로써 데이터베이스 접근 수단을 제공한다. 결국, 이 함수들은 똑같은 코드로 모든 데이터베이스에 접근할 수 있도록 해 준다.

Fig. 2는 구현된 웹 애플리케이션의 구성으로 각 요소들의 역할과 관계를 상세하게 보여주고 있다. 애플릿은 전처리 과정으로 grid(), init() 함수를 포함하며, 후처리 과정으로는 paint()를 포함한다. 결국, 애플릿은 GUI를 통하여 서버와 사용자를 잇는 역할을 하게 된다. 서버는 웹 서버와 서블릿 엔진을 포함하며 주 계산을 수행하고 데이터베이스는 애플릿과 서블릿의 공통 저장소로서 자료의 전달 및 공유를 가능하게 한다.

위의 구성은 적지 않은 문제들을 안고 있다. 먼저, 애플릿의 용량이 크면 브라우저를 통한 다운로드와 프로그램 적재에 많은 시간이 소요된다. 또, 인터넷을 통해 데이터베이스에 대량의 자료들이 저장되거나 추출되므로 시간이 오래 걸

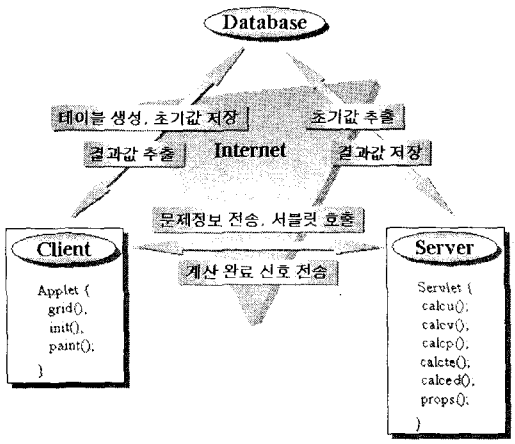

Fig. 2 Implemented architecture using Applet -Servlet communication, JDBC and Swing

리거나 동작하지 않을 수 있으며 애플릿에서 직접 데이터베이스에 접근하므로 자칫 시스템에 부담을 줄 수 있다. 그래서, 본 연구가 진행되는 동안 이러한 문제를 해결하기 위해 많은 노력을 기울였다.

사용자가 해당 홈페이지를 열면 Fig. 3과 같은 GUI를 보게 된다. 좌측 테이블은 격자의 좌표 값과 각 격자점에서의 속도, 압력, 난류 에너지, 소산율 등을 숫자로 표시하고 있으며 처음 실행되었을 때의 값은 초기값들을 보여준다. 하단의 그래픽 영역은 테이블에 표시된 값들을 그림으로 확인할 수 있도록 되어 있으며 계산 완료 후에는 이를 통하여 미리 보기를 수행할 수 있다. 프레임의 우측 영역은 문제를 설정하기 위한 버튼과 입력 창들을 제공하고 있다. 그 밖에 상단의 메뉴 바를 사용하거나 메뉴에 표시된 단축키를 이용해 다른 기능들을 수행할 수 있다.

문제를 설정하고 시작 버튼을 눌러 계산 서블릿을 호출하게 되면 잠시 후 서버 측으로부터 메시지가 전달된다. 이것은 서버에서 계산 과정 동안 보내는 메시지이며 에러 발생 시에는 이를 알리고 정상적으로 계산이 완료되는 경우에는 계산 소요시간을 보내게 된다. 에러가 발생되는 원인은 서블릿의 오작동, 원활하지 않은 통신 상태, 데이터베이스 시스템의 접근불능 등을 들 수 있다. 완료 신호를 받은 사용자는 서버 측 계산 시간과 총 소요시간을 확인할 수 있다.

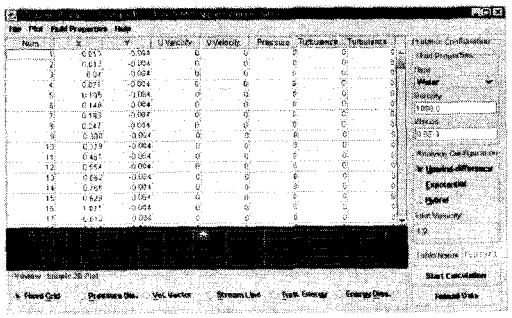

Fig. 3 GUI of the applet for preprocessing

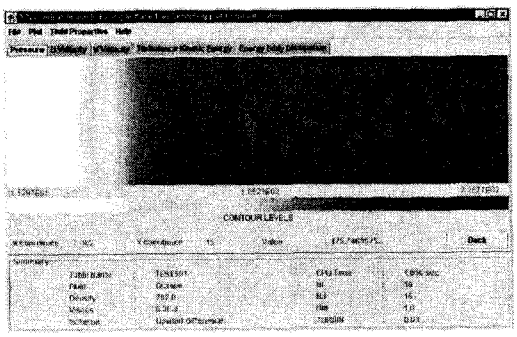

Fig. 4 GUI for the applet for postprocessing

계산 완료 후 데이터 갱신 버튼을 눌러서 서블릿과 데이터베이스 시스템으로부터 결과를 넘겨받으면 사용자는 테이블과 하단의 버튼을 사용해서 미리 보기를 수행할 수 있다. 상단의 메뉴나 단축키를 사용하여 결과보기를 수행하면 Fig. 4와 같은 GUI를 볼 수 있으며 이를 통해 각 화소에 해당하는 좌표 점의 결과를 자세히 확인할 수 있다. 결과 값은 각 화소마다 보간법을 사용해 산출된 값이며 마우스를 원하는 위치에 올려 해당 좌표의 결과들을 확인할 수 있다. 또, 문제에 대한 정보를 데이터베이스에서 얻어 요약란에 보여 주도록 구현하였다.

이외에도 데이터베이스를 활용하여 결과를 저장하거나 불러올 수 있게 하였으며 도움말도 데이터베이스에 저장된 내용을 불러오도록 구현하였다. 또, 데이터베이스에 저장된 유체의 속성을 추가하거나 삭제할 수 있도록 구현하였다. 결국, 데이터베이스의 모든 자료를 여러 사용자들이 공유할 수 있게 하였으며 도움말이나 프로그램의 업그레이드를 서버 측에서 한 번만 작업하도록 해서 인터넷의 장점을 충분히 활용할 수 있도록

하였다.

2.3 구현 결과 검토

예상했던 문제들을 해결하는 것은 그다지 어려운 일이 아니었다. 애플릿의 다운로드 시간을 없애기 위해 미리 다운 받도록 처리하였다. 먼저, 필요한 클래스(class)들을 압축해서 두 개의 파일로 만들었다. 그리고, 항상 떠 있어야 하는 프레임에 이 압축된 파일들을 사용하면서 아무 일도 하지 않는 또 다른 애플릿을 만들어 놓아 다운로드 시간을 줄였다. 또, 데이터베이스에 연결할 때는 전송할 시점에만 연결을 유지하도록 처리해서 시스템에 부담을 주지 않도록 하였다. 걱정했던 데이터저장시의 시간 지연은 없었으며 30개의 문자로 이루어진 수천 개의 데이터가 저장되거나 추출되는데는 단지 몇 초 정도가 소요되었을 뿐이었다.

간단한 예제이기는 하지만 변환에 약 20 시간, 전체 작업에 약 250 시간 정도가 소요되었다. 작업에 큰 장애나 어려움은 없었고 프로그램 실행에도 별 문제가 없었다.

3. 분산프로그래밍으로 구현한 수치해석

3.1 열전도 해석

다중 쓰레드[7]와 자바 RMI(Remote Method Invocation)[8]를 이용한 분산 프로그래밍 구현의 가능성을 확인하기 위해 가장 간단한 예제를 사용했다. 역시 포트란으로 작성되어 있으며 초기에 100K로 균일하게 유지되고 있는 표면이 t=0인 시점에 x=0과 x=L에서 급작스럽게 0K로 냉각되는 1차원 열전도 현상을 해석하고 있다. 지배방정식은 다음과 같다.

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \qquad (3)$$

여기서 α는 열 확산계수이다.

3.2 분산프로그래밍 구현

추후 규모가 큰 작업을 위한 기초를 마련하기 위해 Fig. 5에서처럼 전처리 과정과 후처리 과정을 생략하고 파일전송(File Transfer)으로 대신했다. 여기서 애플릿이 하는 역할은 사용자가 서

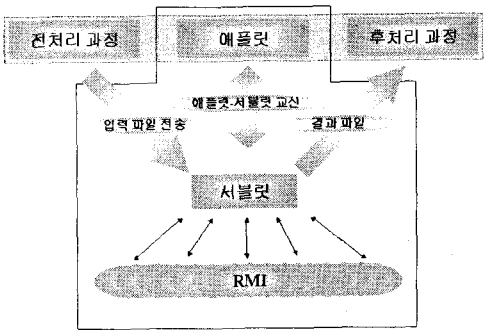

Fig. 5 Implemented architecture using Multi-Threads, Java RMI, file transfer and Applet-Servlet communication

버에서의 계산과정을 확인할 수 있도록 하는 것이다.

소스를 자바로 변환하고 서블릿으로 만들었으며 이 서블릿은 다중 쓰레드(Multi-Thread)를 구현한 객체를 생성한다. 이 클래스(Class)는 포트란의 do 문에 해당하는 자바의 for 문을 원하는 수만큼의 쓰레드에 분배할 수 있도록 만들었고 자바의 재사용성을 바탕으로 하여 제작하였기 때문에 이 클래스를 상속한 새로운 클래스를 만들고 함수를 재정의(overriding)하는 방법으로 손쉽게 다른 작업에도 적용할 수 있다. 만약 서블릿이 작동되는 서버의 중앙처리장치(CPU)가 두 개 이상이라면 이 부분까지의 구현만으로도 이 구성의 이득을 볼 수 있을 것이다.

본 연구에서는 각각의 쓰레드에서 RMI 서버의 함수를 호출하도록 했다. 이 함수들은 실제 for 문의 안에 있는 계산 부분이 구현되어 있다. RMI는 자바에서 제공하고 있는 네트워크 분산프로그래밍 기술이다. 네트워크 분산 프로그래밍은 먼 곳에 위치한 시스템에 있는 객체의 함수를 마치 같은 시스템에 있는 객체를 사용하듯 동일한 구문으로 호출하도록 한다. 이렇게 네트워크 상에서 객체를 자유롭게 분산시킬 수 있는 것을 분산 객체 구조라 한다. 분산 객체를 사용하는 CORBA(Common Object Request Broker Architecture)는 복잡한 프로토콜을 사용하고 필요 이상의 많은 기능들을 지원하도록 설계되었

다. 또, 지원하는 미들웨어가 무겁고 관련 기술이 빠르게 변하고 있어 구현이 어렵다는 단점이 있다. 분산 객체를 사용하는 또 다른 하 DCOM(Distributed Common Object Model)으로 시스템에 제약이 있다는 단점이 있다. 반면, RMI는 비교적 구현이 쉽고 시스템에 대한 제약도 없으며 자바의 다른 모든 기능들을 개발에 활용할 수 있게 한다. 자바 객체만을 지원한다는 단점이 있기는 하지만 변환이 그렇게 어려운 경우가 아니라면 충분히 고려해볼 만한 가치가 있다고 생각한다.

3.3 구현 결과 검토

본 연구에서는 공통의 저장소가 필요 없도록 경계가 명확할 수 있는 for문만을 적용시켰다. 그러나 대부분의 수치해석 for문들은 경계를 명확히 구분할 수 없다. 이는 RMI 객체와 통신할 필요가 없다면 쉽게 공통의 저장소를 확보할 수 있으므로 결코 문제가 되지 않는다. 따라서 여러 개의 CPU에 계산을 분산하는 경우에는 아주 쉽게 작업이 이루어질 수 있다. 그러나 RMI 객체를 사용하게 되면 서로의 결과를 공유해야 할 공통의 저장소를 구성하는 일이 결코 쉬운 일이 아니다. 이는 추후의 연구에서 해결해야 할 문제이며 아마도 데이터베이스나 객체간 통신을 사용하게 될 것이다. 또 여기에서는 로드밸런싱을 전혀 고려하고 있지 않으므로 통신에 문제가 생기거나 계산을 수행 중인 시스템 성능에 문제가 발생하면 프로그램 전체는 작동하지 않게 되거나 서버에 쓸데없는 문제를 일으킬 수 있다. 이 역시 추후의 연구에서 비중 있게 다루어져야 할 문제이며 효율적인 객체간의 통신이 논의되어야 할 것이다.

4. 결론

인터넷은 분명 우리에게 많은 변화를 가져다 주었다. 그러나, 현재의 서비스들은 그 실제 능력을 제대로 발휘하고 있지 못하며 이러한 현상은 당분간 지속될 것이다. 특히, 기계공학 분야에도 인터넷이 활용될 수 있는 여지들이 많이 있음에도 불구하고 일부 특정 관련분야를 제외하고는 적용된 경우가 거의 없는 실정이다.

이에 본 연구에서는 간단한 예제를 통하여 기계공학의 한 분야인 수치해석에 인터넷의 장점을 활용할 수 있는 가능성을 확인하였으며 개발 수단으로 자바를 소개하고 하나의 구현 모델을 제시하였으며 간단한 두 예제를 통하여 제시된 구성의 구현 가능성을 확인함으로써 추후 실질적이고 진보적인 연구를 위한 기반을 마련하였다.

여기에 그치지 않고 보다 고급의 인터넷 기술들을 활용한 연구가 이어져서 큰 규모의 수치해석 ASP(Application Service Provider)가 개발되었으면 하고 아울러 다른 기계공학 분야에도 인터넷 관련 기술이 효율적인 역할을 할 수 있었으면 하는 바람이다.

참고문헌

(1) Danny Ayers 외 14 공저, 하수정 역, 2000, "Java Server Programming", 정보문화사. pp. 323~337.

(2) 아라까와 츠우이치(あらかわ つういち) 저, 명현국 역, 1997, "수치유체공학", 한미출판사, pp. 163~232.

(3) Patankar, S.V., 1980, "Numerical heat transfer and fluid flow", Hemisphere, pp. 126~131

(4) Kathy Walrath 외 1 공저, 류광 역, 2000, "The JFC Swing Tutorial", 정보문화사.

(5) Jason Hunter 외 1 공저, 최환진 역, 1999, "JAVA 서블릿 프로그래밍", 한빛미디어.

(6) George Reese 저, 조국, 유호경 역, 1998, "JDBC와 JAVA로 데이터베이스 프로그래밍 하기", 한빛미디어.

(7) Scott Oaks 외 1 공저, 진장일 외 1 공역, 2001, "자바 쓰레드", 한빛미디어, pp. 279~321.

(8) QUSAY H. MAHMOUD 저, 곽용재 편역, 2001," "자바를 사용한 분산 프로그래밍, 인포북", pp. 178~243.