韓國精密工學會 2002年度 秋季學術大會論文集 pp.492~496 KSPE 02F058

분산 객체 기술을 이용한 웹 기반 기계 모니터링 시스템

차주헌*(국민대 기계자동차공학부), 공호성**(KIST 트라이볼로지 연구센터)

Web-Based Machine Mornitoring System Using Distributed Object Technology

Joo-Heon Cha and Hosung Kong

Abstract

We present the web-based remote monitoring system using distributed object technology. In order to provide the desired functionality, the system has used CORBA(Common Object Request Architecture) and Java Servlet to implement the integrated distributed object environment. It converts the existing standalone machine monitoring system into web-based machine monitoring system. It consists of applet program, CORBA server and CORBA client. The usefulness of our system will be illustrated by the application to ICM(Integrated Condition Monitoring) System developed by KIST Tribology Center.

Key Words: CORBA, Java, Servlet, Diagnostics, Web-enabled engineering, Remote monitoring

1. 서론

최근 각종 시스템에서 정보화에 관한 많은 논의가 진행되고 있다. 산업현장에서도 예기치 않은 사고를 예방하고 사고를 미연에 방지하기 위한 모니터링 시스템의 필요성이 대두되고 있다. 그리고, 인터넷의 성능이 개선되면서 지리적, 공간적인 제약을 벗어나 산업 전반에서 인터넷을 기반으로 한 정보시스템의 구축이 진행되고 있다.

그러나, 기존 시스템과의 플랫폼, 개발 언어 등의 비호환성은 새로운 시스템을 도입하는데 많은 어려움으로 작용하고 있으며 , 실제로 시스템을 구축하였다고 하더라도 구축 후에 수시로 변하는 시스템 로직에 따른 유지, 보수의 어려움과 확장성에서는 많은 개선점이 필요하다. 특히, 기존에는 사용자 인터페이스 부분과 시스템 로직 부분이 결합되어 있었기 때문에, 둘 중에 한 부분이라도 변경이 생기면 새로 구축하는데 많은 시간이 소모되었다. 인터넷은 공용망이기 때문에 동시접속자가 늘어날 경우 시스템의 성능저하가 눈에 띄게 나타나게 된다. 또한, 시스템에 불순한 동기를 가진 클라이언트의 접속에 의한 위험으로부터 시스템을 보호해야 하는 부분도 고려해야 한다.

본 논문에서는 기존 시스템의 통합과 분산객체 환경을 구축하기 위하여 OMG(Object Management Group)의 표준인 CORBA(Common Request Broker Architecture)를 적용시키고, JAVA Servlet을 이용하여 UI(User Interface)부분과 시스템 로직을 분리하는 시스템 구조를 제시한다.

2. 적용 기술

2.1 CORBA

CORBA는 OMG에서 제정한 분산 객체 기술의 표준으로서 하드웨어와 운영체제에 구애받지 않으며 사용이 가능하고, 다른 언어로 작성된 프로그램 사이에서 서로에게 객체를 전달 할 수 있으며, 객체의 메소드 호출도 가능하다.

Fig.1은 CORBA 클라이언트/서버 시스템에서 클라이언트 객체가 ORB(Object Request Broker)라는 소프트웨어 버스를 활용해 원격지 서버의 메소드를 호출하고, 이의 수행 결과를 주고받는 것을 간단하게 도식화한 것이다. 여기서 ORB는 중개자 역할을 하게 되는데, 객체 사이의 상호작용을 조절하고, 객체의 위치에 관계없이 접근할 수 있는 위치 투명성을 제공한다.

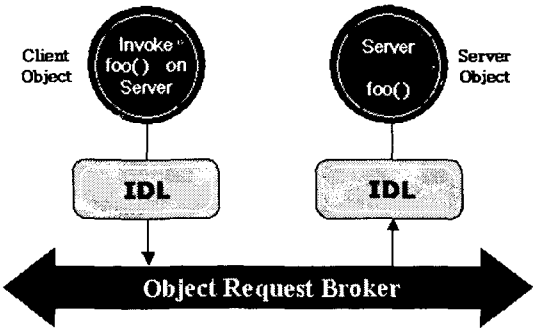

Fig.1 CORBA model

CORBA 클라이언트와 서버 객체간의 통신에 있어서 외부에 노출된 서버의 인터페이스는 CORBA 표준 인터페이스 언어인 IDL(Interface Definition Language)로 표현되며, IDL로 표현한 인터페이스의 내용은 IDL 컴파일러를 이용하여 JAVA, C++ 등의 언어로 변환해 바인딩 함으로써 CORBA가 서로 다른 언어의 한계를 뛰어넘을 수 있게 된다.

만약 분산객체 기술을 사용하지 않는다면 그 모든 module들을 일일이 source merging을 통하여 하나의 프로그램으로 구성하고 컴파일/링크를 해야 하는 일련의 복잡하고 시간이 많이 걸리는 작업을 수행해야 한다. 또한 이런 방법은 각각의 module들이 같은 프로그래밍 언어를 사용하고 있어야 가능한 일이다. 하지만, CORBA를 사용하게 되면 각 module들이 어떤 프로그래밍 언어를 사용했는지에 관계없이 IDL code만 작성하여 기존의 module을 CORBA 객체로 쉽게 wrapping할 수 있다[1-4].

또 다른 CORBA의 중요한 특징은 기존 시스템에 대한 연동 방안을 제시한다는 것이다. CORBA를 이용함으로써 기존 시스템에 변경을 거의 하지 않고 시스템이 제공하는 API를 이용하여 새로운 응용을 작성하여 비교적 쉽게 시스템을 통합할 수 있다.

또한 CORBA를 통해 개발자는 클라이언트와 서버 사이의 통신에 따른 개발 부담을 줄일 수 있다. 기존 시스템 연동이라는 CORBA의 장점을 사용함으로써 복잡한 시스템을 객체 지향적으로 쉽게 구축할 수 있고, 다양한 기능의 Web클라이언트를 제공할 수 있다[5].

2.2 Java Servlet

Servlet은 서버의 기능을 확장하기 위해 동적으로 적재될 수 있는 자바 클래스로서 Fig.2에서 보듯이 웹서버와 연동 하여 웹브라우저를 통해 서비스를 제공하는 역할을 하며 일반적으로, 기존의 IIS(Internet Information Service) 나 Apache등의 웹서버와 연동하여 HTTP 프로토콜을 통하여 사용자의 요청을 처리하며, 다중 접속자 환경을 지원한다[6-8].

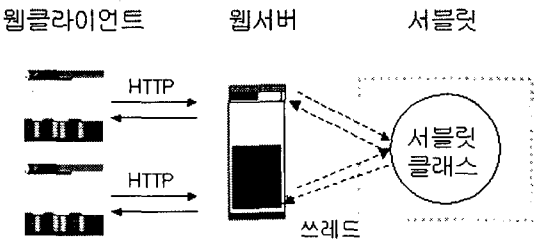

Fig. 2 Java Servlet

Servlet은 자바의 클래스이기 때문에 기본적으로 멀티-쓰레드를 지원하며 다중 접속자 환경에서 요청당 프로세스를 생성하는 CGI(Common Gate Interface)스크립트에 비해 서버에 부하가 줄어든다. 아울러, 자바가 지원하는 객체 지향언어의 장점과 자바 API가 지원하는 각종 라이브러리를 모두 활용할 수 있다[9].

사용자 입장에서는 기존의 인터넷 서비스와 동일한 인터페이스를 제공하며, 서버측에서는 프로그램을 통한 동적인 HTML코드 생성이 가능하기 때문에 확장성이 용이하다.

마지막으로, Servlet은 다른 자바 서버측 기술들과의 상호 연계가 용이하여 J2EE(Java2 Enterprise Edition) architecture, EJB(Enterprise Java Beans) 등과의 연동이 가능하다. 또한 Servlet-Applet연동도 가능하며, JDBC(Java Database Connectivity)API로 인하여 각종 DBMS(Database Management System)와 연동을 수행할 수 있다[10].

3. 시스템 구성

본 시스템은 KIST의 Tribology 연구센터에서 개발한 통합 기계 건강진단 시스템인 ICM System(Integrated Condition Monitoring System)을 CORBA와 JAVA Servlet을 이용하여 분산 객체기반의 클라이언트/서버 시스템으로 구현한 것이다. Fig.3은 이와 같은 전체 시스템의 개념도를 보여 준다.

기존 ICM System은 기계의 측정장비에 연결된 현장 컴퓨터에서 실행되며 각 부위의 마멸, 온도, 진동 등을 일정 시간 간격으로 측정하고 이렇게 측정된 data를 Database에 저장시킨 후 화면에 수치와 그래프로

디스플레이를 하여 사용자가 손쉽게 기계의 상태를 파악할 수 있도록 되어 있다. Fig.4는 ICM의 실행 모습을 보여준다.

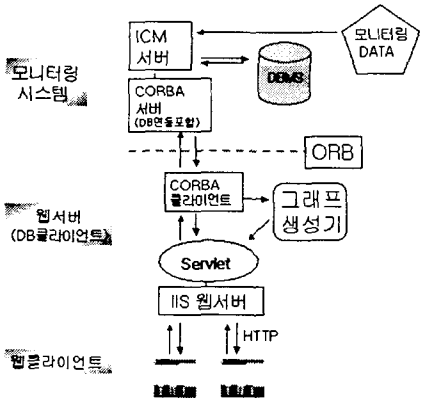

Fig. 3 System structure

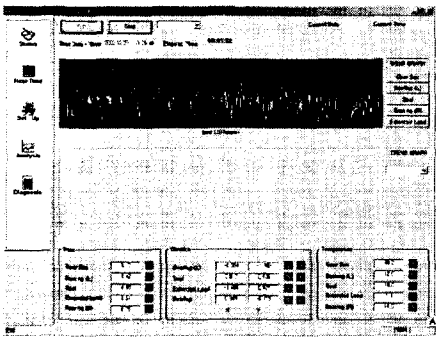

Fig. 4 Example of ICM system

CORBA의 장점 중의 하나는 기존의 독립 모듈을 CORBA 인터페이스로 wrapping하여 분산객체화 시킬 수 있다는 것인데, 이런 장점을 기존의 ICM System에 적용하여 CORBA Server로 wrapping하고 JAVA Servlet기술을 적용시켜서 클라이언트에게 결과를 보여준다.

이처럼 기존 시스템을 분산 객체 기반 클라이언트/서버 시스템으로 구축해 놓으면 사용자는 웹 브라우저를 통하여 간편하게 모니터링 작업을 수행할 수 있다.

본 논문에서 제안한 시스템은 다음과 같이 크게 3개의 part로 분류할 수 있다.

3.1 CORBA Server

기존 시스템을 CORBA 인터페이스로 wrapping 하여 객체화시킬 수 있는 분산 객체 시스템의 장점을 활용하여 ICM System을 CORBA Server로 wrapping 할 때 Fig.5에 나타나 있는 것처럼, ICM System의 원래 기능 중에서 data 검색과 graph data 모듈 등을 분리하여 IDL로 정의하고, 독립적인 구현객체를 만든다. 이 구현객체를 통해서 클라이언트들의 서비스 요청을 수행하도록 개발한다. 기존 ICM System과 CORBA Server를 위해 wrapping한 부분은 서로 완전히 독립적으로 구현되어져 있으며 이와 같이 module의 독립성을 최대로 유지시키고 기존 ICM 부분과 CORBA 연결/실행 부분은 서로 독립적인 thread로 구성하여 나중에 다른 module을 추가하거나 기존 module을 수정할 때에 편리하도록 구성하였다.

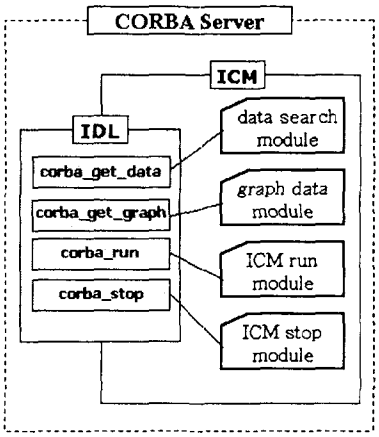

Fig. 5 Wrapping by CORBA

또한, CORBA Server는 동시에 여러 클라이언트들에게 서비스를 제공해야 하므로 멀티 스레드 시스템이 되어야 한다. 하지만, 클라이언트/서버 시스템에서 둘 이상의 클라이언트가 동시에 하나의 Database에서 같은 record를 access하여 입력, 출력, 수정 등의 작업을 할 때 충돌이 일어나게 되는 이른바 동기화 문제가 발생할 수 있다. 이런 동기화 문제를 해결하기 위해서 MFC(Microsoft Foundation Class)의 Multi thread 동기화 객체들을 사용하였다. 즉, 하나의 클라이언트가 특정 Database의 record에 access를 시도할 때 동기화 객체를 이용하여 해당 record를 Lock하고 작업이 끝난 후 UnLock을 해서 다른 클라이언트의 해당 record의 access를 허용한다.

3.2 CORBA Client

CORBA의 ORB는 서버측 분산객체들의 시스템로직 변화에 따른 확장성을 담당하며, Fig.6에서 보듯이, CORBA 클라이언트는 별도의 자바 클래스로서 Servlet과 CORBA 서버와의 중간 계층을 형성한다. 또한 CORBA 클

라이언트는 ORB를 통해 C++로 구성된 분산객체와의 통신에서 가교역할을 하기 때문에 사용자 인터페이스, 사용자 인증 등의 Servlet의 변경을 CORBA 서버와 독립적으로 수행할 수 있다.

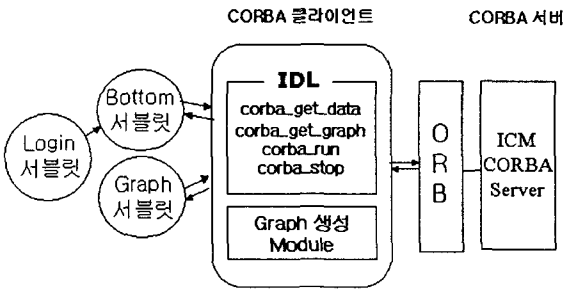

Fig. 6 CORBA Client

3.3 Servlet을 이용한 웹 연동

자바 서버측 기술인 Servlet을 이용하여, 사용자는 웹브라우저만으로 원격 모니터링 작업을 수행할 수 있도록 구성되어 있다.

여기서 Servlet은 CORBA 클라이언트와 연동하여, 사용자로부터 요청을 받아들이고 이를 CORBA 클라이언트에 전달해주며, CORBA 클라이언트로부터 오는 결과치를 동적인 HTML생성을 통하여 사용자에게 뿌려준다.

기존의 CORBA-Web 연동 모델인 Applet-CORBA 모델이나 CGI-CORBA 모델 등에 비해 Servlet-CORBA 모델은 많은 장점들이 있는데, 가령 CGI가 사용자의 요청이 있을 때마다 프로세스를 하나씩 생성함에 반하여, Servlet은 사용자의 요청에 따라 쓰레드를 발생시키므로 동시 접속자가 늘어나도 시스템의 리소스를 적게 차지한다. 또한, Servlet은 서버측에서 실행되면서 사용자에게는 단지 결과만 보내주기 때문에 필요한 소스를 다운 받아야만 하는 애플릿보다 실행 속도가 훨씬 빠르다.

본 시스템은 분산 환경에서의 웹 모니터링 시스템으로서 다음과 같은 사항을 고려하여 Servlet을 설계하였다. 첫째로, 동시 접속자 처리와 사용자 인증 등의 부분을 Servlet에 위치시킴으로써, 모니터링 시스템의 확장과 사용자 인터페이스의 확장을 독립적으로 수행할 수 있도록 구성하였다. 둘째로, Servlet과 JDBC를 연동하여 CORBA를 통하지 않고 별도의 사용자 데이터베이스의 정보를 통해 사용자 인증을 하도록 구성하였다. 마지막으로, 사용자의 요청은 HTTP프로토콜을 사용하고 Servlet이 사용자의 요청을 중계하기 때문에 사용자가 서버에 직접 접근함으로써 발생하는 보안 문제를 제거하였다.

또한, 클라이언트의 요청에 의하여 Database에 저장된 데이터를 그래프로 만들어서 보여주는 기능도 제공한다. 기존 Applet을 이용하는 시스템인 경우, 그래프 출력에 필요한 모든 데이터를 사용자의 웹브라우저까지 전달해야 하고, 그래프 출력 부분 역시 클라이언트에서 수행되어야 하므로 시간이 많이 소요되고 클라이언트에게 부담이 큰 것이 사실이었다. 하지만 본 시스템에서는 그래프 생성을 위한 데이터가 CORBA Server에서 CORBA Client인 자바클래스까지만 이동하고, 여기서 GIF(Graphics Interchange Format) 파일형식의 그래프 이미지가 만들어진다. 따라서 사용자에게는 서블릿을 통하여 단지 GIF 파일형식의 그래프 이미지만 전송된다[11,12]. 따라서 사용자에게 보내는 데이터 크기가 대폭 줄어들어 속도가 향상되었으며, 클라이언트의 부담도 크게 줄어들게 하였다. Fig.7은 이와 같은 그래프 생성 방법을 보여준다.

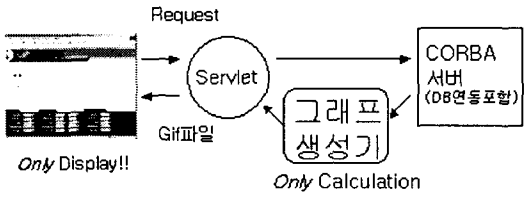

Fig. 7 Generation of graph data

4. 개발 사례

본 시스템은 Windows 2000 server 와 Windows 98 운영체제에서 구현되었으며, CORBA 사용을 위하여 IONA사의 Orbix 3.0.1 과 OrbixWeb 3.1c을 이용하였다. 또한, IIS 5.0으로 웹서버를 구축하였고, Servlet API로써 JSDK 2.0을 사용하였고, 웹서버와 Servlet의 연동에는 Resin 1.2.0을 이용하였다.

우선, 분산객체 환경을 위하여 CORBA Daemon을 실행시키고 CORBA Server Manager로 ICM Server를 구현객체로 등록시킨 후 ICM CORBA Server를 실행시키면 이제부터 CORBA Client의 요청을 받아들일 수 있는 상태가 된다(Fig.8 참조).

Resin을 실행시켜서 Servlet을 실행하면 클라이언트들이 웹 브라우저로 접속 가능한 상태가 된다(Fig.9 참조).

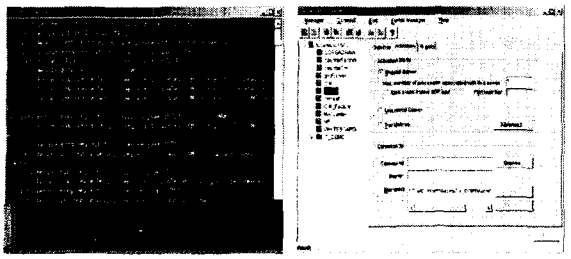

Fig. 8 CORBA daemon and server manager

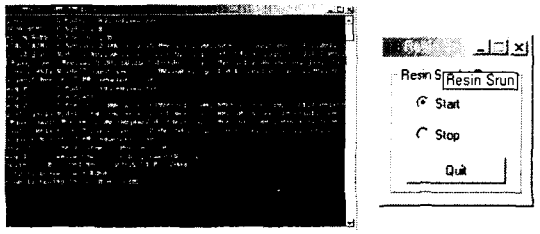

Fig. 9 Example of Resin

Fig.10은 여러 클라이언트가 웹브라우저로 서버에 연결하여 다양한 그래프를 요청하여 원격 모니터링 작업을 수행하는 것을 보여준다. 서버는 클라이언트에게 최근 측정값을 일정 시간간격 마다 보여주고 클라이언트에게 그래프 데이터 요청이 들어오면 필요한 data를 읽어서 그래프 정보를 그림파일로 만들어 클라이언트에게 보내고 Servlet엔진에서 HTML파일을 업데이트 시켜준다.

5. 결론

본 논문에서는 독립 실행형의 기계 모니터링 시스템을 분산 객체 기반의 클라이언트/서버 시스템으로 변환시킴으로써 사용자가 원격지에서 인터넷을 통하여 간단히 모니터링 작업을 수행할 수 있도록 하였다.

또한 CORBA를 이용한 분산 객체 시스템은 각 객체의 Language나 OS(Operation System)에 구애받지 않기 때문에 확장성이 아주 뛰어나서 이후에 다른 기능의 모듈을 추가할 때 간편하게 작업할 수 있다. 그리고 JAVA Servlet을 이용한 웹서버를 통하여 CORBA의 객체들과 연동하여 클라이언트에게 서비스를 제공하는 방법은 클라이언트 쪽에 아주 간단한 인터페이스를 제공하고 부담을 최소화시킬 수 있다.

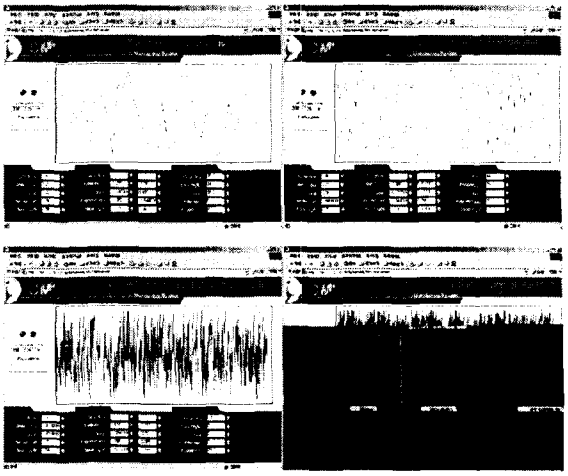

Fig. 10 Machine monitoring system by Internet

6. 참고문헌

1. Reaz Hoque, "CORBA3", 정보문화사, 2000.
2. Robert Orfali, Dan Harkey, Jeri Edwards, "Instant CORBA", WILEY, 1997.
3. 왕창종, 이세훈, "Inside CORBA3 프로그래밍", 대림출판사, 1999.
4. 문왕식, "클릭하세요 CORBA 프로그래밍", 대림, 2000.
5. 서범수, 한태창, 정석찬, 주경준, "자바 Servlet과 CORBA를 이용한 효율적 Web 응용 시스템 구축 기술", 한국정보과학회 제26회 춘계 학술발표논문집, 1999.
6. Jason Hunter, "Java Servlet Programming", O'reilly, 1998.
7. Merlin Hughes 외, "Java Network Programming 2/e", 인포북, 1999.
8. "Java Servlet Specification version 2.3", SUN microsystems, 2000.
9. Elliotte Rusty Harold, "자바 네트워크 프로그래밍, 한빛미디어, 1997.
10. SUN, "Developing Java Web Server with Java Servlets SL-310", SUN Microsystems, 1999.
11. Danny Ayers외, "Professional Java Server Programming", wrox, 1999.
12. Ken McCrary, "Create dynamic images in Java Servlets", http://www.javaworld.com/javaworld/jw-05-2000/jw-0505-servlets.html, 2000.