웹 서비스를 이용한 다중 서버 관리 시스템 설계

홍석건° 김정희, 곽호영
제주대학교 대학원 컴퓨터공학과
{hancem°, carina, kwak}@cheju.ac.kr

Design of Multiple Server Management System Using Web Services.

Seok-Gun Hong° ,Jeong-Hee Kim, Ho-Young Kwak
Dept. of Computer Engineering, Cheju National University

요 약

현재의 서버 관리에서는 직접 접근을 통한 서버 관리 방법이 주를 이루며 서버의 일부를 사용하는 사용자들에게는 접근하기에는 어려운 부분이 많다. 이를 개선하기 위해서 본 논문에서는 웹 서비스를 이용하여 다중 서버 관리 시스템의 구조를 제안한다. 제안 시스템을 적용하면 다수의 서버의 상태를 동시에 파악할 수 있으며 웹 서비스를 이용하였기 때문에 방화벽 내부에 있는 서버에 대한 관리도 가능하다. 또한 인증 과정을 통해 사용자마다 권한을 다르게 부여할 수 있도록 하여 서버의 특성에 맞는 관리가 가능하도록 설계 하였다.

1. 서 론

웹 서비스는 XML 표준을 기반으로 개발된 표준화된 XML 메시지를 통해 네트워크 상에서 접근 가능한 연산들의 집합을 기술하는 인터페이스[1]를 지칭하는 용어로 기존의 웹 환경을 이용한 분산 컴퓨팅을 가능케 함으로써 웹을 통한 시스템 통합을 용이하게 하며, 기존의 분산 컴퓨팅 모델인 CORBA, Java RMI, DCOM 등을 대체할 수 있는 새로운 모델로 주목받고 있는 기술이다. 이런 웹 서비스의 장점으로는 크게 단순성, 상호 운용성, 개방형 표준으로 인한 구축비용의 저렴, 관련 업계의 지원으로 인한 빠른 발전 등을 들 수 있다. 이는 HTTP와 XML을 기본으로 한 기술이기 때문에 나타나는 것이라고 할 수 있다.[2]

웹 서비스를 구현하기 위해 현재까지 표준화된 기술로는 서비스간의 메시지 전달을 위한 메시지 형식을 정의한 SOAP(Simple Object Access Protocol)[3], 웹 서비스 이용에 필요한 자원 및 입/출력 메시지 형식을 기술하는 WSDL(Web Services Description Language)[4], 웹 서비스에 대한 디렉토리 서비스 지원을 목적으로 개발된 레지스트리 표준인 UDDI(Universal Description, Discove-ry and Integration)[5]가 있다. 이외에도 보안(WS-License, WS-Security), 프로세스 흐름(XLANG), 트랜잭션 관리(BTP, 확장 트랜잭션), 메시징(WS-Inspection, WS-Referral, WS-Routing, BEEP, Reliable HTTP) 기술 등 웹 서비스를 보완하기 위한 기술들이 등장하고 있다.

현재의 서버 관리 방법은 관리자가 telnet, SSH(Secure Shell) 또는 특정 서버에 종속적인 서버 관리 프로그램(Server Management Application)을 통한 직접 접근을 통해서만 관리가 가능하다. 한명의 서버 관리자가 다수의 서버를 관리해야 하는 경우에는 각각의 서버에 접속을 해야 하며 서버의 가동상태를 동시에 확인할 수 있는 방법이 없으며 이기종간의 서버를 관리하기 위해서는 각 서버의 운영체제(Unix, Linux, Windows Server 등)에 대한 지식을 별도로 습득해야 하는 등 많은 불편함이 발생한다.

본 논문에서는 다수의 서버를 관리하면서 발생하는 불편함을 보완하며 상호 운용성을 극대화하기 위해 웹 서비스를 적용한 서버 관리 시스템을 제안한다.

본 논문은 2장에서 제안 시스템의 기본 구조를 기술하며, 3장에서는 제안 시스템의 각 부분에 대한 기능 및 구현 방안에 대하여 기술하였다. 4장에서는 본 논문의 결론 및 향후 연구에 대하여 기술하였다.

2. 제안 시스템의 기본구조

서버 관리는 크게 서버의 운영체제와 운영 방법에 따라 그 방법이 다르다.

운영 체제에 따라서는 Unix/Linux 계열과 Windows 계열로 나눌 수 있는데 Unix/Linux 계열인 경우에는 Telnet과 SSH를 이용한 콘솔 접속을 통한 관리가 주를 이루고 있다. 콘솔 접속을 통한 관리 방법은 Unix/Linux 명령을 직접 입력을 통해 처리하는 방식이기 때문에 일반 사용자들이 접근하기는 어렵다. Windows 계열의 서버를 관리하기 위해서는 보통 로컬 관리와 특정 어플리케이션을 이용한 관리가 주를 이루고 있어 관리자 위치의 제한을 준다.

운영 방법에 따라서도 관리 방법의 차이점이 나타나게 된다. 웹 호스팅 서버인 경우에는 사용자 관리, 접속 로그 확인, 트래픽 감시 등의 관리위주이며 데이터베이스 서버인 경우에는 서버 자원에 대한 관리, 방화벽이나 네임서버인 경우에는 각 설정 변경 등의 관리를 하게 된다.

그림1은 현재 대부분의 서버 관리 모델을 나타내고 있다.

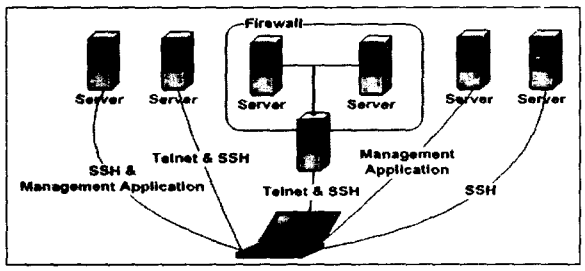

그림 1 기존 서버 관리 방법

본 논문에서 제안하고 있는 서버 관리 시스템은 플랫폼 독립적이며 프로그래밍 언어에 중립적인 웹 서비스를 적용하여 설계 되었다. 즉 웹 서비스의 표준인 SOAP을 이용하여 관리자와 서버간의 처리 명령 및 결과를 전달하며 WSDL을 이용해 서버 정보를 UDDI에 등록한다. UDDI는 서버들의 정보들을 저장하고 있으면서 서버 관리자의 요청에 따라 정보를 제공하게 된다. 이런 모든 통신은 HTTP를 통해서 이루어지므로 특수한 경우를 제외한 방화벽 내부의 서버에 대해서도 특정 애플리케이션 없이 손쉽게 관리가 가능하게 된다.

다음의 그림 2는 웹 서비스를 이용한 서버 관리의 기본 구조를 표현하였다.

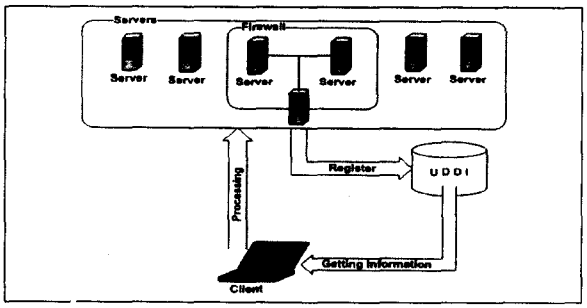

그림 2. 웹 서비스를 이용한 서버 관리 기본구조

각각의 서버에서는 기동 시 서버의 정보를 UDDI에 등록하도록 하며 종료 시에는 UDDI에 등록된 정보를 변경 또는 삭제하도록 하여 서버의 상태를 나타낼 수 있도록 한다. 또한 서버에 대한 작업 요청 시 서버 관리자 인증 확인 후 작업을 처리하도록 하며 세션(Session)연결이 종료될 때까지 인증을 유지하도록 한다.

서버 관리자는 UDDI에 등록된 정보를 통해서 관리 서버들의 현재 상태를 파악하여 서버 관리를 위한 작업을 할 수 있는 정보를 얻을 수 있다.

3. 제안 시스템 설계

제안 시스템은 그림 2의 기본구조를 보완하여 그림 3과 같이 크게 Server 내부에 데몬(Demon)형태로 설치되는 Server Part Application과 Middleware Part로 구성한다.

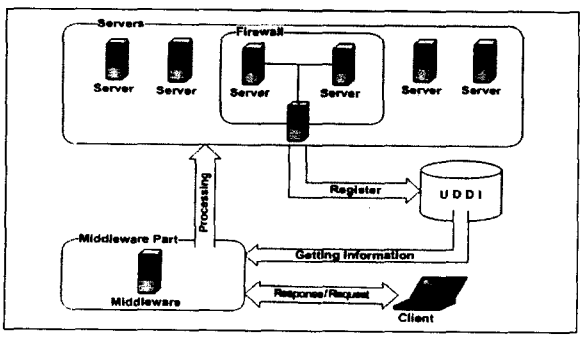

그림 3 제안 시스템 구성

3.1 Server Part Application

Server Part Application의 기본 기능은 서버 관리를 위한 웹 서비스를 개시하며 서버 정보와 서버의 구동 상태를 UDDI에 등록하고 인증 받은 사용자의 요청 명령을 처리하도록 한다.

그림 4에서처럼 Server Part Application은 서버에 대한 직접적인 명령을 처리하는 Server Side Command Processer, 서버 관리 명령들을 웹 서비스로 개시하는 Web Services Processer, 이 중간에 요청 명령을 서버에서 실행시키기 위해 서버 운영체제의 명령으로 변환하는 Command Processer, 명령 실행에 대한 로그를 처리하는 Log Processer, 마지막으로 관리자 인증 부분을 처리하는 Authentication Processer로 구성된다.

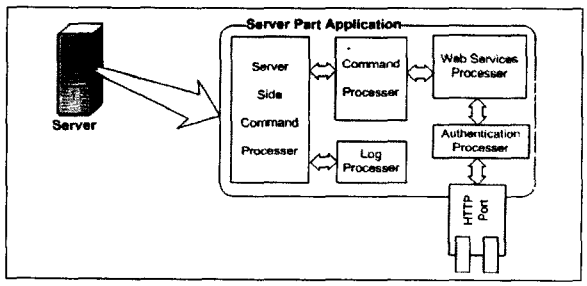

그림 4 Server Part Application의 구조

서버에 대한 접근을 제한하기 위해 Authentication Processer에서는 Middleware Part에 의한 연결에 한해 제한하며 Session에 대해서도 하나로 제한하여 관리의 중복을 방지하도록 하고 Server Part Application 자체의 사용자 인증 처리를 하므로 서버 관리의 단일화와 익명 사용자에 대한 접근을 방지하게 한다.

명령 처리 부분은 서버의 운영체제에 따라 같은 명령일지라도 내부 처리 절차가 다를 수 있기 때문에 Command Processer에서 모듈(module)화하여 명령의 추가/수정/삭제가 쉽도록 하며, 또한 모듈들을 조합할 수 있도록 하여 효율적인 명령 처리가 가능하도록 한다.

Server Side Command Processer에서는 서버와 실제적으로 명령을 처리하는 부분으로서 Command Processer에서 요청한 명령을 실제로 수행하는 부분이다.

서버 관리에서 로그는 서버에 문제가 발생했을 경우 원인을 분석할 수 있는 중요한 목록이다. 따라서 명령 요청이나 요청 종료 시 정보를 Log Processer에 전달하여 로그를 생성하여 저장하도록 한다. 로그에서 저장할 정보는 명령 요청자, 실행 명령, 실행시간, 실행 결과이며 추가적으로 요청자 접근IP등의 요청자 정보도 포함할 수 있다.

3.2 Middleware Part

Middleware Part는 UDDI정보의 무분별한 유출을 차단한다. 또한 인증된 사용자만을 관리 서버에 접근할 수 있도록 UDDI를 통해 얻은 서버들의 정보를 관리자에게 제공하며 관리자의 명령 요청을 해당 서버에서 처리하도록 명령을 전송하는 인터페이스(Interface) 역할을 함으로서 서버 위치에 상관없이 관리자가 접근할 수 있도록 위치 투명성을

제공하는 역할을 한다.

또한 서버 관리자외 제3자에 의한 관리 명령 수행을 방지하며 명령 수행의 신중함을 위해 Middleware Part에서의 관리자 인증 확인은 Server Part와의 관리자 인증과는 별도로 관리하도록 한다. 그리고 하나의 서버는 한명의 관리자만 관리하도록 하여 서버 관리의 중복을 제거한다.

또한 Server Part와 같이 관리 작업에 대한 사항에 대하여 로그를 생성하도록 한다. 이는 서버 관리상에 문제가 발생하였을 경우 Server Part의 로그와 비교 분석하여 문제점을 발견하고 이를 해결하기 위한 방법으로 이용하기 위함이다.

3.3 기타 사항

서버관리의 가장 큰 중점은 보안일 것이다. 즉 서버측에서 사용자에 따라서 실행할 수 있는 명령어의 종류가 다르며 시스템에 치명적인 명령도 실행시킬 수 있다. 하지만 웹 서비스는 XML 기반의 SOAP를 이용해 메시지를 HTTP를 통해 전송하기 때문에 정보가 노출될 가능성이 매우 높다. 이를 위해 XML 전자서명(XML Digital Signature)[6]과 XML 암호화(XML Encryption)[7], XML 기반의 공개 키 관리 명세인 XKMS(XML Key Management Specification)[8] 등을 통해 정보 노출을 차단하며 승인 정보의 안전한 교환을 위한 SAML(Security Assertion Markup Language)[9]과 접근 제어 리스트를 통해 보안이 요구되는 자원에 대한 미세한 접근 제어 서비스를 제공하는 XACML(XML Access Control Markup Language)[10]를 이용하여 서버 관리시 안전성을 확보하도록 한다.

또한 SOAP를 통해 전송되는 자료의 양이 방대해 지는 경우를 위해 SOAP 메시지를 GZIP를 이용하여 압축[11]하여 전송함으로서 메시지의 보안과 전송의 효율성을 증대시키도록 해야 할 필요성이 있다.

4. 결 론

본 논문에서는 기존 다수의 서버 관리의 불편함을 보완하면서 플랫폼 독립적이며 프로그래밍 언어에 중립적인 웹 서비스를 이용한 서버 관리 시스템을 설계하였다.

제안한 서버 관리 시스템의 이점으로는 웹 서비스를 이용함으로서 플랫폼에 종속적이지 않으며 어플리케이션 개발시 효율성을 증대시킬 수 있다. 또한 Middleware를 이용함으로서 서버의 위치 투명성을 증대 및 서버의 보호 능력 또한 증가시킬 수 있다.

하지만 서버의 운영 체제 및 플랫폼에 따라 Server Part 내부적 수행부분의 제작의 중복이 발생하여 개발 시 효율성이 저하 문제와 HTTP를 이용한 웹 서비스에서의 메시지 전송 방식인 SOAP의 암호화의 완벽한 표준화의 미비로 인한 보안 문제가 발생하게 된다.

보안 문제는 3.3 기타사항에서 기술된 바와 같이 현재 보안관련 기술[6,7,8,9,10]들의 연구가 진행되고 있으며 현재까지 표준화된 기술이 미비하다. 하지만 기술들의 표준화가 완료되면 서버 관리상의 보안 문제는 대부분 해결될 것이라 본다.

향후 연구는 관련 기술을 적용한 제안된 시스템의 구현에 있다.

참고 문헌

[1] Header Kreger, IBM Software Group, "Web Services Conceptual Architecture(WSCA 1.0)", http://www-4.ibm.com/software/solutions/webservices/pdf/WSCA.pdf, May 2001.

[2] 이경하, 이규철, 웹 서비스의 표준화 동향과 발전 방향, 데이터베이스연구회지,VOL.19 NO.01 pp. 0080~0087, 2003.03

[3] Martin Gudgin, et al., "SOAP Version 1.2 Part 1: Messaging Framework", http://www.w3.org/TR/soap12-part1, Recommendation, W3C, June 2003.

[4] Roberto Chinnici, et al., "Web Services Description Language Version 2.0 Part 1: Core Language", http://www.w3.org/TR/wsdl20", Working Draft, W3C, November 2003.

[5] Ariba Inc., Microsoft Corp. "UDDI Technical White Paper", http://www.uddi.org/pubs/Iru_UDDI_TechnicalWhite_Paper.PDF, uddi.org, September 2000.

[6] XML Signature Press Release, http://www.w3.org/2002/02/xmlsignature-pressrelease.html.en.

[7] XML Encryption, http://www.w3.org/Encryption/2001.

[8] XML Key Management Specification, http://www.w3.org/XKMS.

[9]Security Assertion Markup Language, http://www.oasis-open.org/committees/security.

[10] XML Access Control Markup Language, http://www.oasis-open.org/committees/xacml/index.shtml.

[11] Brian D Goodman, "Squeezing SOAP", http://www-106.ibm.com/developerworks/webservices/library/ws-sqzsoap.html, March, 2003.