분산환경에서의 JINI를 기반으로 한 모니터링 시스템

송무찬[O] 임성훈 김정선
한양대학교 컴퓨터공학과
{mcsong, shim, jskim}@cse.hanyang.ac.kr

Network Monitorring System based on JINI in Distributed Environment(RMS)

Moochan Song[O] Seonghun Im Jungsun Kim
Dept. of Computer Science and Engineering, Hanyang University

요 약

현재의 기업 네트워크 상에는 다양한 분산환경 시스템들이 존재한다. 이와 더불어 업무에 사용되는 서버들이 갈수록 증가하는 추세에 있다. 이러한 다양한 서버들이 존재하는 분산환경 하에서 서버의 시스템 지연 및 단절 사태가 증가하고 데이터의 폭증에 따라 서버의 병목현상이 갈수록 심화되고 있다. 이를 관리하기 위해 많은 비용과 인력이 낭비되므로 통합된 환경에서 다양한 서버들을 관리할 수 있는 시스템이 필요하게 되었다. 따라서, 본 논문에서는 JAVA, JINI, JNI, C 라이브러리를 이용한 중앙 집중식 모니터링 관리 시스템을 제안하고자 한다.

1. 서 론

컴퓨터 기술의 비약적인 발전으로 네트워크 망이 전 세계적으로 급속하게 갖추어지기 시작 하였고, 이와 더불어 네트워크를 통한 서비스 증가로 인해 서비스를 제공하는 서버의 수는 기하급수적으로 증가하게 되었다. 각 서버들은 자체적으로 모니터링을 하는 도구를 가지고 있기는 하지만, 서버들의 증가로 인해 각 네트워크에 분산이 되어 있는 서버들의 상태를 관리자가 로컬에서 일일이 체크하기에는 상당한 인적자원과 비용이 필요하게 되었다. 이러한 상황은 관리자로 하여금 네트워크 상의 분산되어 있는 서버들의 특정 어플리케이션의 자원사용 여부 및 부하 여부등의 서버 상태에 대한 모니터링에 대한 요구를 더욱 증폭시키게 되었다. 본 논문에서는 분산환경 하에서 상이한 서버들을 중앙 집중식으로 모니터링을 함으로써 효과적으로 서버를 관리할 수 있는 시스템의 설계 및 구현을 하였다. 본 논문의 구성을 살펴보면, 2장에서는 RMS 관련기술에 대한 기술이 언급될 것이고, 3장에서는 RMS의 기능을 설명하고, 4장에서는 RMS의 세부구현에 대해서 살펴보고, 마지막 5장에서는 결론 및 연구방향을 살펴볼 것이다.

2. 관련 연구

2.1 JINI

지니는 썬 마이크로시스템즈에서 개발한 미들웨어로써 자바를 기반으로 네트워크 상에 접속된 프린터나 스캐너등의 기기들 혹은 디스크 드라이브와 같은 장치들의 접속이나 공유를 단순화하기 위한 기술이다. 이러한 장치들을 컴퓨터나 네트워크에 추가하려면 설치와 부팅과정이 필요하지만, 지니를 지원하는 장치는 스스로 자기 자신을 네트워크에 알리고 자신의 능력에 관해 자세한 정보를 제공하며, 네트워크 상의 다른 장치들이 즉시 액세스할 수 있도록 한다.[1]

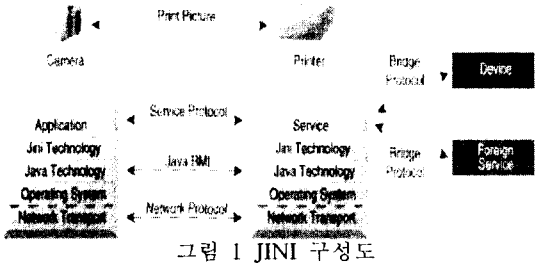

그림 1 JINI 구성도

지니 시스템은 그림 1에서처럼 자바를 기반으로 하기 때문에, 운영체제나 기타 하드웨어 플랫폼에 무관하게 동작하게 된다. 지니의 각각의 구성요소들은 Discovery, Join, Lookup 프로토콜을 이용하여 상호 통신이 이루어지며 서비스 제공하는 부분과 서비스를 사용하는 부분의 상호작용은 자바 RMI(Remote Method Invocation)를 이용한다. 다음으로, 지니 시스템의 수행과정을 살펴보면 첫째, 서비스 제공하는 부분은 Discovery 프로토콜을 이용하여 서비스를 관리하는 관리자를 찾고, 둘째, Join 프로토콜을 이용하여 서비스 관리자에게 서비스를 등록하며, 셋째, 서비스 이용하는 부분은 Discovery 프로토콜을 이용하여 서비스 관리자를 찾고 Lookup 프로토콜을 통해 자신이 원하는 서비스를 검색하여, 부합하는 서비스

를 사용하게 된다.[2][3]

2.2 JNI(Java Native Interface)

현재 자바에서 다른 언어로 작성된 코드를 호출하기 위해 만들어진 규약으로는 C/C++에 대한 호출만을 지원하고 있다. 자바의 처리 속도 문제는 자바가 안고 있는 가장 큰 단점이기도 하다. 하지만 JIT 컴파일러와 같은 기술은 자바 프로그램의 속도를 빨라지게 하고 있지만, 자바는 원칙적으로 바이트 코드를 인터프리트해 수행되기 때문에, 자바의 기술이 발전한다고 해도, 네이티브 코드의 속도를 따라갈 수는 없다. 따라서 아주 빠른 처리가 요구되는 대량의 작업이나 실시간(real-time) 처리에서 자바를 적용시키기는 무리가 있을 것이다. 특히, 일반적인 목적이 아닌 특수한 목적으로 제작된 하드웨어의 경우, 자바 프로그램을 통해서 제어해야 할 필요가 있을 시에는 자바만으로는 해결하기는 힘들다. 그렇기 때문에 JNI의 필요성이 대두된다. 이 JNI는 기존에 작성된 프로그램이나 기존의 시스템(legacy)과 연계를 하는 규약이다. JNI를 적용시키는 주된 이유는 플랫폼에 따라 다르게 제공되는 서비스를 이용할 수 있다는 점이다. 그리고 자바의 클래스 라이브러리는 방대하고 다양한 서비스를 제공하지만, 특정 플랫폼에서 제공하는 고유의 서비스의 기능을 모두 포함할 수는 없다. 따라서, 모든 시스템에서 JNI를 사용해서 프로그램 하는 것은 자바가 지니는 장점들을 해치는 결과를 초래할 수 있지만, 특정 부분에 적절하게 사용한다면 JNI는 자바의 장점과 C/C++의 장점을 골고루 이용할 수 있는 길을 제공한다.[4][5]

2.3 Wrapper Facade Pattern

Wrapper Facade 패턴의 목적은 객체 지향 클래스 인터페이스들을 통해서 로우 레벨의 함수들과 데이터 구조들을 캡슐화 시키는 것이다.[6] 예를 들면 sockets, pthreads, GUI functions같은 네이티브 OS C 언어 API들을 캡슐화 시키는 AWT와 같은 클래스 라이브러리들을 들 수 있다. 만약 네이티브 OS C 언어 API을 이용해 네트워킹 프로그램을 작성하면 복잡하고, 이식성이 떨어지며, 유지보수를 어렵게 만들지만 Wrapper Facade

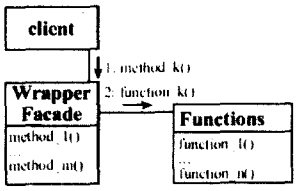

그림 2 Wrapper Facade Pattern UML class diagram

패턴을 이용하면 더 간결하고 강한 프로그래밍 인터페이스를 통해서 어플리케이션의 이식성과 유지 보수성을 개선하는 효과를 가져온다.

3. RMS 기능

RMS는 그림 3과 같이 다수의 에이전트와 하나의 서버로 구성된다. 각 에이전트의 구성요소로서 통신을 담당하는 Communication 모듈, 각 에이전트의 리소스를 수집하는 Monitoring 모듈, 이들을 관리하는 Management 모듈로 구성이 되어 있다. 서버는 JINI를 이용하여 에이전트를 인식하는 에이전트 인식 모듈, 각 에이전트와 통신을 담당하는 Communication 모듈, 모니터링 된 데이터를 저장 및 에이전트를 관리하는 Management 모듈로 구성이 되어 있다.

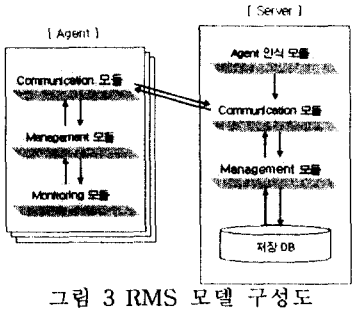

그림 3 RMS 모델 구성도

3.1 Server

서버는 등록 요청을 한 모든 에이전트의 List를 유지하여야 하고 등록된 각 에이전트에 대해 Enable, Disable 시킬 수 있어야 한다. 또한 각 에이전트에 대해 모니터링 하고자 하는 application을 선택하여 어플리케이션 마다 monitoring을 위한 time interval을 설정해서 에이전트들에게 전송한다. 서버는 모니터링 요청을 받은 에이전트들로부터 CPU 사용량, Memory 점유율등의 Resource 사용정보를 비 동기적으로 수신하여 저장한다.

3.2 Agents

에이전트들은 서버의 에이전트 인식 모듈을 통해서 자동으로 서버를 찾아내어 자신의 등록을 요청한다. 등록된 에이전트들은 서버에서 설정된 어플리케이션들에 대한 Time Interval을 통해서 선택된 어플리케이션들의 CPU 사용량, Memory 점유율등의 Resource 사용정보를 수집하고 Time Interval 간격으로 비 동기적으로 서버에게 전송을 한다(Time Interval 설정 요구가 없는 경우 Default 값을 설정한다).

4. RMS 설계 및 구현

4.1 Server

그림 4는 RMS의 서버 class diagram이다. JINI를 이용해서 에이전트들이 서버에 등록이 되면 서버는 에이전트들의 오브젝트들을 만들어 에이전트들의 리스트를 유지한다. 그리고 어플리케이션 리스트를 agent들로부터 전송받아 각 에이전트들에 대한 어플리케이션별 Time Interval을 설정 후 설정 데이터를 저장한다. 서버는 설정된 데이터를 에이전트들에게 전송하여 모니터링을 요청하게 된다. 서버는 모니터링을 시작한 agent들로부터 수집된 데이터를 비 동기적으로 수신을 받아서 DB에 저장하게 된다.

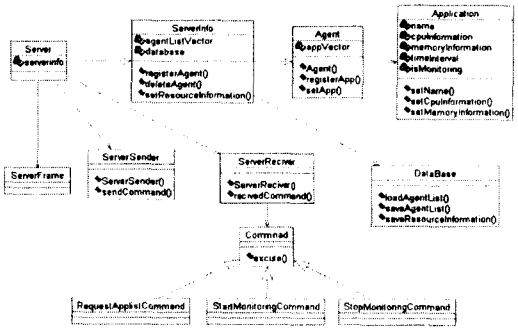

그림 4 RMS server class diagram

4.2 Agents

그림 5는 RMS의 agent class diagram이다. 각 에이전트는 프로그램의 시작과 동시에 서버에 등록을 하게 된다. 서버의 모니터링 요청이 있을 시에 현재 에이전트의 어플리케이션 리스트를 서버에 전송하게 되고, 서버에서 설정된 데이터의 수신을 통해서, 에이전트는 어플리케이션에 대한 모니터링을 시작하게 된다. 이때 모니터링 데이터의 수집을 담당하는 클래스는 Wrapper Facade Pattern을 이용한 클래스로 모니터링 데이타를 수집하기 위해 C 라이브러리 사용하는데, 이 메카니즘은 JNI를 통해서 JAVA와 C 라이브러리를 연결하여 다양한 유닉스 시스템과 OS버전에 따라 손쉽게 확장이 가능하고 이식성을 보장하고 유지보수를 쉽게 만들어 준다.

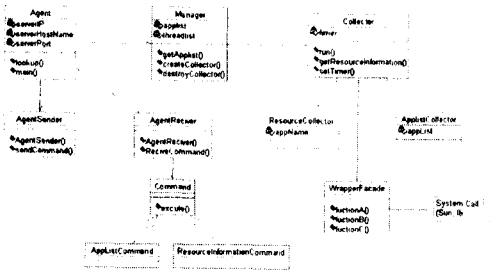

그림 5 RMS agent class diagram

5. 결론 및 연구 방향

본 논문에서는 네트워크 상에 존재하는 다양한 분산환경에서 서버들을 효과적으로 관리하기 위한 실시간 모니터링 시스템을 JAVA, JINI, JNI, C 라이브러리를 이용하여 설계 및 구현을 하였다. 이 시스템은 JNI라는 인터페이스를 통해 구현을 하기 때문에, 다양한 유닉스 시스템 및 OS버전에 따라 손쉽게 확장이 가능하고 이식성과 유지보수를 쉽게 만들어 주고 있다.

향후 본 시스템은 보다 더 다양한 OS로의 적용과 OS에 독립적인 모니터링 데이터를 통해서 시스템 분석툴로의 확장이 가능하다. 또 모니터링 데이터를 통해서 모니터링 시스템 뿐만이 아니라 서버의 진단 및 예방이 가능한 시스템으로 확장이 가능하다고 하겠다.

[참고 문헌]

[1] JINI Tutorial, Jan Newmatch
http://pandonia.canberra.edu.au/java/JINI/tutorial/

[2] SUN Microsystems JINI Network Technology
http://www.sun.com/jini/

[3] W.Keith Edwards "Core JINI", Prentice-Hall, 2001

[4] Grodon, Rob "Essential JNI: Java Native Interface", PH/PTR, 1998

[5] SUN Microsystems JNI Specification
http://java.sun.com/products/jdk/1.2/docs/guide/jni/spec/jniTOC.doc.html

[6] D.Schmidt, M.Stal, H.Rohnert F.buschmann, "Pattern-Oriented Software Architecture", Wiley, 1999