오픈 아키텍처 기반의 이기종 분산 환경에서 애플리케이션 관리 및 모니터링 시스템 구현

민법기*, 국승학*, 김현수*, 김점수**
*충남대학교 컴퓨터공학과
**국방과학연구소
e-mail:bkmin@cnu.ac.kr

Implementation of the Management and Monitoring System for Applications on the Open Architecture based Heterogeneous Distributed Environment

Bup-Ki Min*, Seunghak Kuk*, Hyeon Soo Kim*, Chum-Su Kim**
*Dept. of Computer Sc. & Eng., Chungnam National University
**Agency for Defense Development

요 약

본 논문에서는 오픈 아키텍처 기반의 이기종 분산 시스템에서 애플리케이션 관리 및 모니터링 시스템을 구현한다. 전투관리 시스템이나 공정관리 시스템 등과 같은 통합관리 시스템은 다양한 이기종 분산 시스템으로 구성되며 이러한 분산 환경에서 다양한 애플리케이션들이 동작하게 되는데 이들을 통제 감시하는 방법이 필요하다. 본 논문에서는 추상적인 정보모델을 이용하여 이기종 시스템들을 표현하고 이를 통해 상호의존성을 관리함으로써 다양한 애플리케이션을 효과적으로 관리하고 모니터링 할 수 있는 방안을 제시한다.

1. 서론

네트워크의 확산과 더불어 여러 이기종의 시스템의 통합 기술의 발전을 통해 분산 시스템이 보편화되기 시작했다. 기존의 분산 시스템은 소켓과 RPC(Remote Procedure Call) 기능을 이용하여 개발되었고 이 후에는 DCE(Distributed Computing Environment)등의 통합 분산 환경을 통해 개발되었다. 그러나 기존의 이러한 방법은 분산 환경 하에서 다양한 하드웨어와 소프트웨어, 운영체제, 데이터베이스 등을 일관되게 연결해 주지 못했다. 이러한 상황을 해결하기 위하여 OMG(Object Management Group)에서는 CORBA(Common Object Request Broker)와 DDS(Data Distribution Service) 등 이기종 분산 환경 시스템을 위한 미들웨어 표준들을 제정하였다[1].

오픈 아키텍처는 데이터 통신과 컴퓨터 네트워크의 OSI(Open System Interconnection)의 개념과 마찬가지로 시스템 제작 시 표준화된 규격을 따름으로써, 시스템의 교체 및 최신기술의 유연한 대처를 가능하게 하는 구조이다. 이를 통해 시스템의 각 장치들이 제작사나 종류에 종속되지 않고 시스템 내부에서 투명하게 결합된다[2].

그러나 시스템의 규모가 커지면서 다양한 이기종 시스템의 통합에 대한 요구사항은 날로 증가하고 있다. 이에 따라 통합 시스템에 포함되어 있는 시스템들을 관리하고 모니터링해야 할 필요성 역시 증가하고 있다. 이러한 통합 시스템에 새로운 시스템이 추가된다면 추가되는 시스템의 특성에 따라 관리 및 모니터링 방법이 달라질 수 있다. 같은 기종으로 이루어진 환경의 시스템은 이를 통합하거나 관리하는 것이 쉽게 이루어질 수 있으나 서로 다른 기종으로 이루어진 환경은 시스템이 늘어감에 따라 시스템을 통합하여 관리하는 것이 쉽지 않다. 따라서 이러한 다양한 이기종 환경에서 시스템의 관리 및 모니터링을 할 수 있는 방안이 필요하다.

본 논문에서는 전투관리 시스템이나 공정관리 시스템 등과 같은 이기종 환경의 분산 시스템을 위한 오픈 아키텍처 기반의 관리 및 모니터링하는 시스템을 구현한다. 본 논문에서는 이기종 시스템들과 여기서 동작하는 애플리케이션들에 대해 추상화된 모델을 사용하여 표현함으로써 시스템과 애플리케이션들의 의존성을 쉽게 관리할 수 있다. 또한 새로운 시스템이 도입되더라도 쉽게 추가할 수 있으며, 이 시스템에 대한 관리 및 모니터링도 효과적으로 수행할 수 있다.

2. 관련연구

분산된 컴퓨터들의 효율적인 감시와 제어를 위해 분산 시스템에서의 관리 및 모니터링 방법에 대한 연구가 활발히 진행되고 있다[3].

[3]의 논문에서는 분산시스템에서의 호스트들을 실시간으로 관리 및 모니터링하여 무분별한 네트워크의 사용을 막고 보다 효율적인 사용 환경을 위한 모니터링 시스템을 제안하였다. 그러나 [3]의 경우에는 동일한 환경의 분산 시스템에서의 관리 및 모니터링을 하는 시스템으로 이기

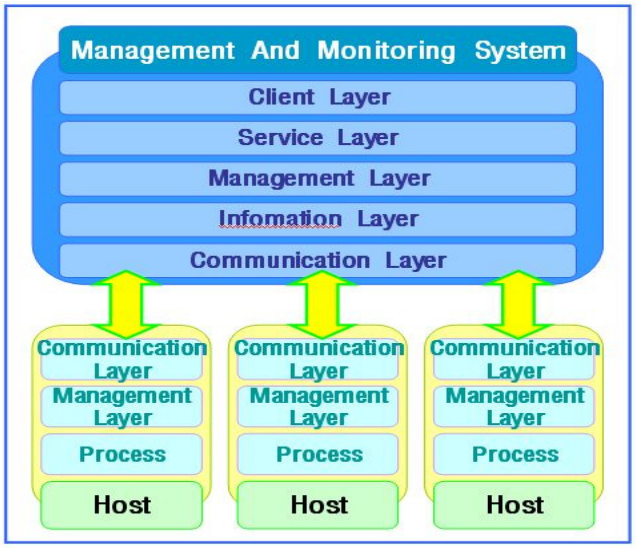
종 분산 시스템에는 적합하지 않는 구조이다.

[4]의 경우에는 이기종간 분산된 시스템들의 시스템을 관리하고 부하분산을 고려한 실시간 시스템을 제안하였다. 하지만 이기종간 분산된 시스템들을 어떻게 통합관리 하는지에 대해서는 설명이 명확하지 않으며 폐쇄적인 구조로 설계 및 구현하였다.

[5]의 경우 네트워크 내의 이기종 운영체제를 가진 개인 컴퓨터의 자원 사용률을 모니터링 하는 시스템을 제시하고 있다. 이 논문에서는 문제가 발생한 컴퓨터를 검출하고 부하분산을 돕기 위한 성능 평가 결과 값을 일정시간 마다 갱신하고 모니터링 정보를 그래프를 통해 제공하여 준다. 그러나 이 논문의 경우에는 컴퓨터를 모니터링만 할뿐 관리기능은 제공하지 않는다.

그러나 본 논문에서는 오픈 아키텍처를 기반으로 통합 관리 시스템을 구현하여 어떠한 종류의 시스템에서도 적용할 수 있을 뿐만 아니라 재사용성과 비용절감에도 효과를 볼 수 있다. 또한 추상적인 정보모델을 이용하여 구현하였기 때문에 어떠한 이기종 분산 시스템에서도 상호의존성을 고려한 애플리케이션 관리 및 모니터링을 수행할 수 있다.

3. 관리 및 모니터링 시스템 구현



(그림 1) 통합관리 시스템의 아키텍처

이기종 분산시스템을 위한 통합관리 시스템의 아키텍처는 그림 1과 같다. 관리 및 모니터링 시스템은 관리 대상이 되는 각각의 호스트들에게 명령을 전달할 수 있으며 현재 호스트들의 시스템 상태를 체크하여 사용자에게 보여줄 수 있는 기능을 제공한다.

각각의 호스트들은 실제적으로 프로그램들이 실행되고 그 실행정보와 자신의 시스템 정보를 관리 및 모니터링 시스템으로 전달하여 사용자에게 정보를 제공한다.

3.1 애플리케이션 및 하드웨어 정보 모델 계층

실질적인 관리 대상이 되는 하드웨어 및 소프트웨어 시스템에 대한 정보 모델들이다. 본 논문에서는 관리 대상이 되는 시스템과 실제 관리 시스템에서의 정보모델을 분리하여 관리하고 각각의 독립성을 보장한다. 또한 이기종 환경의 소프트웨어들의 추상적인 모델을 이용하기 때문에 각각의 환경에 영향을 받지 않고 소프트웨어나 하드웨어 시스템의 관리 및 모니터링이 가능하다.

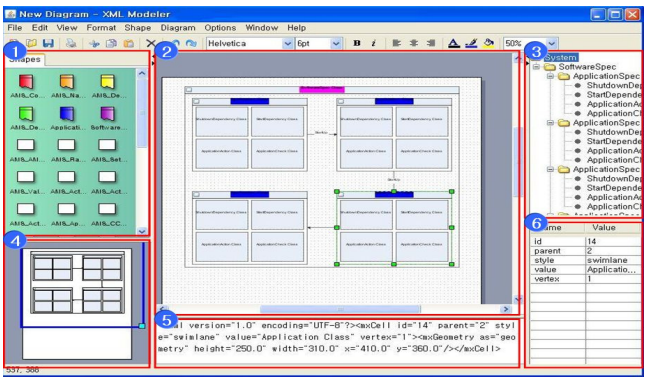
3.2 관리 및 서비스 계층

관리 시스템에서는 크게 애플리케이션의 상태를 모니터링하고 관리하는 기능과 하드웨어의 상태를 모니터링하고 관리하는 기능을 제공한다. 애플리케이션들은 서로 간에 의존성을 가지고 있다. 하드웨어는 동작하고 있는 도중에 멈추게 된다면 자신이 하는 일을 누군가가 대신 해야만 한다. 관리 시스템에서는 상태정보를 모니터링을 통해 파악하고 다른 하드웨어에서 대신 수행할 수 있게 한다.

3.3 정보모델 모델러

관리 대상이 되는 호스트, 소프트웨어 시스템 및 애플리케이션들에 대한 추상적인 정보 모델은 XML로 기술된다. 이 XML문서를 보다 쉽게 작성할 수 있는 도구가 정보모델 모델러이다. 정보모델 모델러는 블록을 이용하여 관리 대상 호스트, 소프트웨어 시스템과 시스템 내의 애플리케이션 등을 표현할 수 있으며, 이들 간의 의존성과 각각의 특성 등을 표현할 수 있다.

3.3.1 정보모델 모델러의 구성



(그림 2) 정보모델 모델러 UI

정보모델 모델러는 크게 6가지의 기능창으로 나누어진다. 각각의 기능창이 하는 역할은 다음과 같다.

① 관리 및 모니터링 시스템에서 정보모델을 생성하는데 필요한 모든 클래스들이 정의되어 있는 창이다.

② 정보모델의 구조를 직접적으로 구현하는 창이다. ①번 창에서 클래스를 선택하고 드래그하여 클래스를 추가할 수 있다.

③ ②번 창에서 만들어진 정보모델의 구조를 트리형태로 확인할 수 있다.

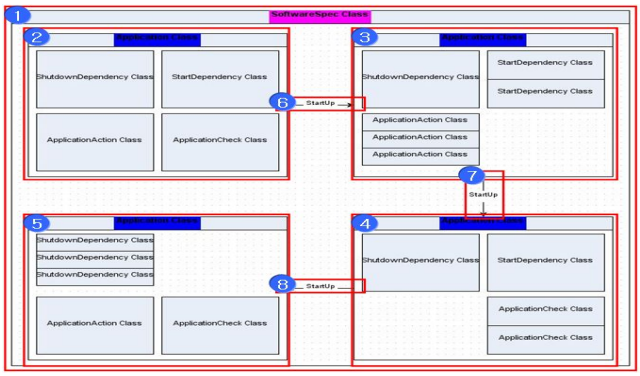
④ ②번 창에서 그려진 정보모델의 전체적인 구조를 보여주는 창이다.

⑤ ②번 창에 그려진 정보모델이 XML 코드로 변환되어

보인다. 그려진 정보모델이 가지고 있는 속성 값들을 적용하여 XML 코드가 작성되어 사용자에게 보여준다.

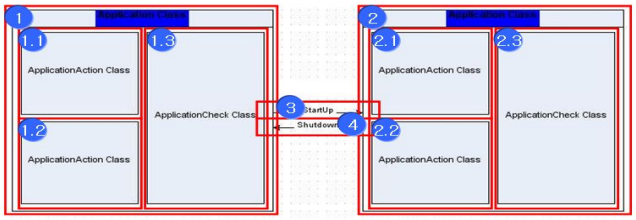
⑥ ②번 창에 그려진 정보모델 요소의 속성을 변경할 수 있는 창이다. 각 속성의 값을 수정함으로써 모델 요소의 속성 값을 설정해 줄 수 있다.

3.3.2 모델러를 이용한 정보모델링의 예



(그림 3) 소프트웨어 및 애플리케이션 정보모델링

그림 3에서는 모델러를 이용하여 ①번의 소프트웨어 시스템의 논리적인 정보모델을 작성하였다. 이 소프트웨어 시스템은 ②~⑤번까지 4개의 애플리케이션으로 구현되며, 이들은 실행명령어, 실행정보, 실행 및 종료 의존성 등을 가지고 있다. 각각의 애플리케이션들마다 가지는 내부 정보는 조금씩 다르지만 담당하는 역할은 동일하다. 이 애플리케이션들 마다 가지고 있는 의존성은 ⑥~⑧번처럼 화살표로 나타내어 주며 이들을 이용하여 하나의 시스템에 있는 것처럼 사용할 수 있다.



(그림 4) SMAS 실행을 위한 정보모델

그림 4는 J2EE환경에서 EJB(Enterprise Java Bean)의 컨테이너 역할을 하는 SMAS(Sun Microsystems Application Server)와 리눅스 환경의 MySQL의 의존성을 나타낸 정보모델 예제이다.

(1) J2EE 환경의 SMAS의 정보모델

(1.1) SMAS의 실행 명령어가 모델링되어 있는 ApplicationAction Class로 SMAS가 실행할 때 사용된다.

(1.2) SMAS의 종료 명령어가 모델링되어 있는 ApplicationAction Class로 SMAS가 종료할 때 사용된다.

(1.3) SMAS의 실행환경 정보가 모델링되어 있는 ApplicationCheck Class로 실행 시 J2EE의 환경에서 구동 가능하다는 조건을 가지고 있다.

(2) 리눅스 환경의 MySQL 정보모델

(2.1) MySQL의 실행 명령어가 모델링되어 있는 ApplicationAction Class로 MySQL을 실행할 때 사용한다.

(2.2) MySQL의 종료 명령어가 모델링되어 있는 ApplicationAction Class로 MySQL을 종료할 때 사용한다.

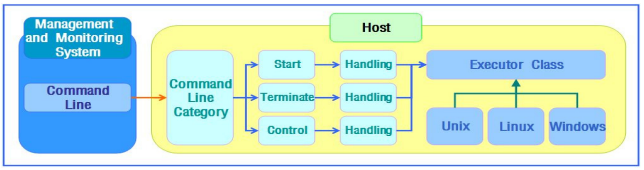
(2.3) MySQL의 실행환경 정보가 모델링되어 있는 ApplicationCheck Class로 실행 시 리눅스환경에서 구동이 가능하다는 조건을 가지고 있다.

(3) SMAS와 MySQL의 실행 의존성을 표현한다. 이로 인해 SMAS가 실행되기 전에 MySQL이 실행되고 있는지를 관리 시스템으로부터 확인을 한 후 MySQL이 실행된 후에 SMAS가 실행된다.

(4) SMAS와 MySQL의 종료 의존성을 표현한다. 이로 인해 MySQL이 종료되기 전에 SMAS가 종료가 되었는지를 관리 시스템으로부터 확인을 한 후 SMAS가 종료된 후에 MySQL이 종료된다.

3.4 호스트측 관리계층의 구현

호스트에서는 관리 시스템으로부터 전달받은 명령을 바탕으로 자신이 가지고 있는 프로그램 및 프로세스들을 관리하고 상태를 모니터링한다. 호스트는 실행 중인 모든 프로세스의 PID를 저장하여 관리한다. 또한 관리 시스템을 통해서 생성되어 실행되고 있는 프로세스만 관리해야 하며 프로세스가 생성되고 종료되는 경우, 호스트에서 관리하고 있는 PID를 갱신해야한다.



(그림 5) 호스트의 명령어 실행 방법

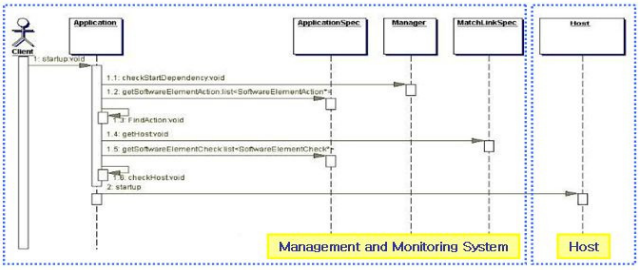
관리 및 모니터링 시스템에서 전달 받은 명령어를 호스트의 명령어로 판별해주는 Command Line Category에서 명령어를 받는다. Command Line Category에서는 전달받은 명령어의 종류를 구분하여(예: 스타트 명령어, 제어 명령어 등), 각 호스트의 운영체제 환경에 맞게 추가적인 핸들링을 한다. 만약 프로세스의 생성이나 종료와 관련 있다면 호스트는 프로세스의 정보를 업데이트하기 위해 관리하고 있는 PID 목록을 갱신한다. 이렇게 핸들링된 명령어는 호스트의 운영체제 환경에 따라 실행방법이 다른데, 이를 해결하기 위해 Executor Class로부터 상속받아 각 운영체제 환경에 맞는 클래스들을 생성한다. 이 클래스들에 핸들링된 명령어를 전달하고, 이 명령어를 해당 클래스에서 실행한다.

3.5 애플리케이션 관리 및 모니터링 동작

3.5.1 애플리케이션 관리 및 통제

관리 및 모니터링 시스템에서는 다음과 같이 애플리케이션을 실행시킨다.

(1) 사용자는 관리 및 모니터링 시스템에서 구현된 UI를 통해 애플리케이션을 선택하고 실행버튼을 누른다.



(그림 6) 애플리케이션 실행 방법

실행 명령어가 Application으로 전달되고 먼저 실행되어 있어야 할 애플리케이션이 있는지를 확인하기 위해 ApplicationSpec으로부터 다른 애플리케이션과의 의존관계 정보를 받아온다.

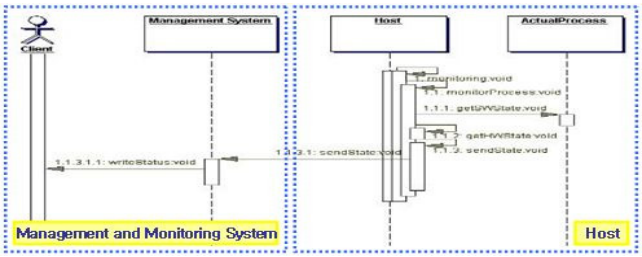
(1.1) 받아온 다른 애플리케이션과의 의존 정보가 존재한다면 Management로부터 먼저 실행시켜야하는 애플리케이션의 정보를 확인하고 동작하지 않고 있다면 해당 애플리케이션을 먼저 실행시키고, (1.2) Application은 다시 ApplicationSpec으로부터 실행 명령어를 모두 받아온다.

(1.3) 받아온 명령어들에서 실행 할 명령어를 찾는다.

(1.4) MatchLinkSpec을 통해 실행될 호스트의 정보를 얻어오고 (1.5) 실행가능한 호스트를 찾는다.

(1.6) 실행이 가능하다면 호스트측에 실행 명령어를 전달하여 애플리케이션을 실행시킨다.

3.5.2 애플리케이션 모니터링



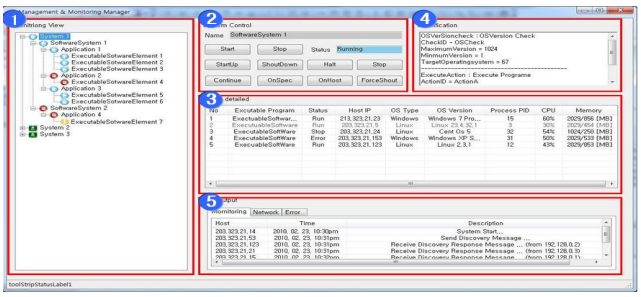
(그림 7) 애플리케이션 모니터링 방법

모니터링을 하기 위해서는 호스트에서 실제 구동하는 애플리케이션들의 상태정보가 필요하며, 호스트들은 상태정보들을 수집하여 관리 시스템으로 주기적으로 전송한다. 관리 시스템에서는 호스트들로부터 전달 받은 정보들을 이용하여 정보모델들을 업데이트하고 사용자 UI로 업데이트된 내용들을 전달하여 사용자에게 최신정보를 제공한다.

3.6 사용자 인터페이스 계층

사용자 인터페이스는 사용자가 관리 및 모니터링 시스템에 연결되어 있는 각각의 호스트들의 상태를 모니터링하고 명령어를 전달하여 관리할 수 있는 프로그램이다.

①Monitoring View는 시스템에서 현재 동작하고 있는 모든 시스템들을 표시한다. 각 관리 대상의 상태를 아이콘으로 표시하여 현재 시스템들의 상태를 한 눈에 알아볼 수 있다. 시스템을 관리하기 위해서는 Monitoring View에서 보여주는 다양한 시스템 중 하나를 선택하면 시스템에 대한 제어 기능을 ②System Control View에서 제공한다. 또한 ③Status Detailed View를 통해 관리 및 모니터링 시스템에서의 다양한 시스템들의 상태정보와 어떤 호스트에서 동작하고 있는지에 대한 정보들을 상세히 볼 수 있다. ④Specification은 소프트웨어 시스템의 상세 스펙 정보를 보여준다. ⑤Log을 통해서는 동작 중에 발생하는 이벤트들에 대한 로그와 프로그램에 대한 에러정보를 확인할 수 있다.



(그림 8) 관리 및 모니터링 UI

4. 결론

통합관리 시스템에서는 일반적으로 다양한 시스템들의 상황을 개별적으로 모니터링하고 제어한다. 본 논문의 시스템에서는 관리대상이 되는 이기종 시스템들의 상호의존성을 고려하였다. 이에 실제 관리대상이 되는 정보모델과 실제 관리 시스템에서의 정보모델을 분리하여 실제 시스템을 독립적인 방법으로 관리 및 모니터링 할 수 있도록 구현하였다. 또한 관리하는 계층과 서비스 계층을 분리하여 사용자가 원하는 새로운 기능을 추가 및 제거를 할 수 있어 다양한 조합의 기능을 제공하는 것이 가능하다.

사사

본 연구는 국방과학연구소의 지원으로 수행되었습니다. (계약번호 UD090017KD)

참고문헌

[1]김홍윤 외 4인, “자바와 코바를 이용한 분산 어플리케이션”, 정보처리, 제4권, 제6호, pp 96-104, 1997

[2]김용팔, “한국전력 서울지역 급전소 SCADA 시스템의 분산/개방형 구조”, 한국자동제어학술회의논문집, pp 1147-1154, 1993

[3]윤치영, 정천복, 황선명, “분산 네트웍 환경에서 실시간 모니터링시스템(NetCop)의 설계 및 구현”, 한국정보과학회, Vol.28. No.2, pp 454-456, 2001

[4]이종대, 구용완, “분산된 서버 관리를 위한 실시간 모니터링 설계 및 구현”, 한국 인터넷 정보학회, 9권 1호, pp 69-78, 2008

[5]P. Domingues, L. Silva, J. G. Silva, “DRMonitor - A Distributed Resource Monitoring System”, Proc. of the 11th Euromicro Conf. on Parallel, Distributed and Network-Based Processing, 2003