능동형 클러스터 관리 시스템의 메타모델 설계

이동훈°, 민덕기
건국대학교 컴퓨터 정보통신공학과
{podong°, dkmin}@konkuk.ac.kr

Metamodeling in Design of Proactive Cluster Management Systems

Donghoon Lee°, Dugki Min
School of Computer Science and Engineering, Konkuk University

요 약

ALBM(Adaptive Load Balancing and Management)은 S/W L4 스위치를 포함하는 능동형 클러스터 시스템이다. 이 클러스터 시스템은 확장 가능한 인터넷 서비스와 적응형 부하분산 처리 능력을 제공한다. ALBM 클러스터 시스템을 설계할 때, 우리는 Model-Driven Development Method를 사용하여 메타모델을 설계하였다. 본 논문에서는 클러스터의 구성을 위한 에이전트관리, 정보관리와 같은 기능과 결함내성을 위한 이벤트관리, 알고리즘관리와 같은 기능을 고려하는 메타모델을 제시한다. 이 메타모델의 초점은 클러스터의 구성관리와 결함관리를 클러스터 관리 시스템의 설계에 반영하여, 새롭게 클러스터 시스템을 설계할 때 쉽게 이러한 기능을 가질 수 있도록 지원하는 것이다.

1. 서론

현대사회에서 IT는 다양한 분야에서 활용되고 있고, 그 환경은 매우 큰 복잡도를 가지고 있다. 이것은 기존 환경에서 고려되지 않았던 소프트웨어의 구조에 기인한다. 기존의 전통적인 공학환경에서는 소프트웨어 기술이 중요한 부분을 차지하지 않았다. 그러나 기존 환경들의 플랫폼 성능이 높아지고 사용자의 요구사항은 다양해졌다. 이것은 하드웨어 기술만으로는 만족시키기 어려운 환경이 되었다는 것을 의미한다.

기존의 소프트웨어를 설계하는 개발방법론들은 그 기반지식을 소프트웨어 개발 환경에 두고 있다. 이것은 전통적인 시스템 개발환경에 익숙한 개발자들에게 새로운 환경에 익숙해지는 것을 요구하기 때문에 어려운 접근방법으로 생각되고 있다.

메타모델 개발 방법론은 전통적인 설계 지식을 모형화해서 이 지식에 기반한 다른 시스템을 쉽게 디자인할 수 있는 환경을 제공하도록 지원한다. 또한 전통적인 기반지식을 소프트웨어 기반지식으로 변형(Transformation)할 수 있는 해석프로그램(Interpreter)를 제작하는 것을 지원하기 때문에, 전통적 기반지식을 사용해서 설계된 시스템을 쉽게 소프트웨어 분야로 변환할 수 있도록 한다.

최근에 IP 패킷의 부하분산 스위치는 네트워크의 한 부품으로서 많이 사용되고 있다. 그러나 여기에는 단순한 패킷의 분산방식인 Round Robin방식과 컴퓨터의 상대적인 성능을 반영한 Weighted Round Robin반식이 가장 많이 사용되고 있다.

ALBM 클러스터 시스템은 여기에 각 노드의 동작 상태를 고려한 **적응형 부하분산 방식(Adaptive Scheduling)**을 지원한다. 이때 노드의 상태는 통신의 상태뿐만이 아니라 노드에서 서비스중인 프로세스의 상태도 포함한다. 이것은 부하분산 알고리즘에서 단순하게 네트워크와 플렛폼의 결함만을 고려하는 것이 아니라 구체적인 서비스의 상태도 고려하는 것을 의미한다.

ALBM 클러스터 시스템은 적응형 방식을 지원하기 위해서 **알고리즘 컴포넌트 구조**를 가지고 있다. 또한 알고리즘 컴포넌트의 실행을 위해서 조건을 판단하는 조건부 이벤트 컴포넌트 구조를 가지고 있다. 알고리즘 컴포넌트 구조와 이벤트 컴포넌트 구조는 능동형 결함관리기능을 수행하는 기반이 된다.

본 논문에서는 클러스터의 구성관리(Configuration Management)에 대한 메타모델을 제시하고, 결함관리(Failure Management)를 위한 알고리즘 컴포넌트 구조의 메타모델 설계를 제시한다.

2. 관련연구

메타모델 분야에서는 다음과 같은 이슈들이 다루어지고 있다. 첫째는 메타모델을 사용하여 특정 **기반 지식을 표현**하는 분야이다. 그 중 주목할만한 연구는 모발 시스템 환경에서의 컴포넌트에 대한 메타모델에 관련된 연구이다[1]. 이 연구에서 모발 환경내의 디바이스들은 리소스, 코드크기와 같은 제약을 가지고 있다고 가정한다. 이러한 제약을 가지고있는

환경에 적용할 수 있는 컴포넌트의 메타모델을 연구하였다.

둘째는 **모델 변환**에 관련된 연구이다. 모델 사이에서 상호변환에 관련된 분야에 메타모델을 적용하는 것이 그 주요 내용이다[2][3]. 특히 프로그래밍 언어들 사이에서의 변환에서도 적용이 되고있다. 또한 원본 모델의 검증을 위해서 검증 모델로의 변환을 수행하고, 그 구현을 검증하는 연구도 진행중이다[4].

셋째는 기존 모델의 **확장**이다. 이를 위해서는 먼저 기존 모델을 메타모델로 만들어야한다. 그리고 그 메타모델에서 확장할 기능을 추가한다. 이것을 사용해서 기존 모델 방법론에 대한 검증기능을 확장하기도 한다[5].

본 논문에서는 Vanderbilt 대학에서 개발중인 The Generic Modeling Environment(GME)[6]를 사용해서 메타모델을 설계한다. 이 GME에는 MetaGME라고 하는 메타메타모델을 제시하고 있다. 이 메타메타모델은 메타모델을 설계하기 위해서 FCO(First Class Object), Atom, Connection, Model과 같은 요소들을 제공한다. 그리고 또한 주목할 만한 다른 활동으로는 OMG(Object Management Group)의 MOF(Meta-Object Facility)가 있다. MOF는 Model Driven Engineering을 위한 표준으로서 CWM이나 UML의 메타모델을 제공한다. GME에서는 MOF로 설계된 모델을 사용할 수 있는 변환기를 제공하고 있다.

3. ALBM 클러스터 시스템 구조

ALBM 클러스터 시스템은 기본적으로 하나의 네트워크를 공유해야 한다. 그리고 서비스를 제공하기 위한 외부 네트워크가 존재해야 한다. 이런 조건을 만족시키기 위한 ALBM 클러스터 시스템의 구조는 그림 1과 같다.

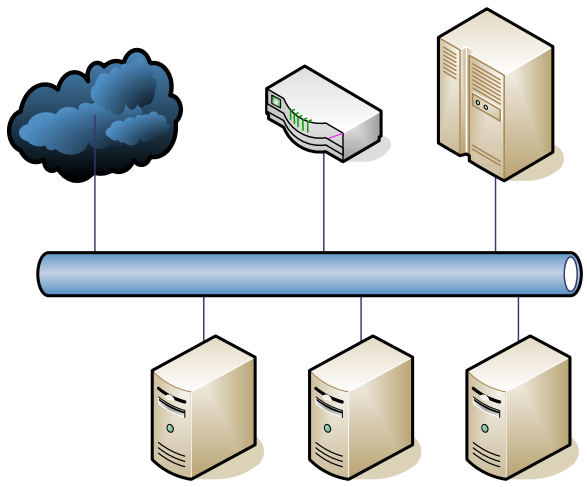

그림 1 - ALBM 클러스터 시스템의 구조

ALBM 클러스터 시스템은 크게 3가지 구성요소를 가지고 있다. 첫째는 Management Station으로 이 요소는 전체 시스템에 대한 관리를 수행한다. 둘째는 Node로 클러스터에 참여하는 컴퓨터들을 나타낸다. 클러스터에 참여하는 모든 컴퓨터에는 Node Agent가 동작하게되고 이 Agent를 통해서 관리가 이루어진다. 셋째는 Director로 Network Packet을 부하 분산 알고리즘에 따라서 각 Node로 전달해주는 기능을 수행한다.

클러스터 시스템의 구성관리 기능은 다음과 같다.

- Discovery
- Alive Checking
- Cluster Configuration
- Election of Coordinator Node

클러스터 구성관리 기능을 사용해서 Station과 Node들이 클러스터를 구성하게 되고 서로를 인식해서 하나의 일을 수행할 수 있게 된다. 또한 클러스터를 구성하고 있는 이웃노드에 대한 정보를 수집하고 동작 검사를 지원한다.

클러스터 환경에는 다양한 종류의 요소들이 참여한다. 그래서 결함상황의 종류도 다양하게 된다. 클러스터 환경에서 발생할 수 있는 결함 상황은 아래와 같다.

- Communication Failure
- Platform Failure
- Agent Failure
- Service Failure

각 결함상황은 플렛폼과 결함이 발생한 장치에 따라서 다양한 형태로 나타나게 된다. 혹은 각 상황은 비슷한 현상으로 나타나게 될 수 있다.

결함상황을 감지하기 위해서 다양한 알고리즘이 시스템에서 실행되어야 한다. 각각의 알고리즘은 시스템의 정보를 수집해서 상황을 판단하게 된다. 사용되는 알고리즘은 다음과 같다.

- Heartbeat
- Detect Failure
- Election

결함상황의 발생과 회복동작에 따라서 주목해야하는 이벤트는 다음과 같다.

- Station is dead
- Station is recovered
- Director is dead
- Director is recovered
- Node is dead
- Node is recovered
- Service is dead
- Service is recovered

위 이벤트가 발생하는것으로 알고리즘이 실행될 수 있고, 그때 클러스터 구성관리 기능이 동작하게 된다.

4. 메타모델 설계

앞에서는 클러스터 시스템의 구성관리와 결함관리를 위해서 필요한 사항들을 설명하였다. 여기에서는 그 사항들을 고려하는 메타모델을 설명한다.

ALBM 클러스터 시스템의 메타모델은 크게 4가지 부분으로 이루어져있다.

- Agent
- Information
- Event
- Active Object

기본적으로 클러스터 시스템은 소프트웨어 Agent으로 구성되어 있다. 모든 클러스터 시스템의 동작은 Agent 사이에서의 상호작용으로 수행된다(Agent View). 각 Agent는 정보를 관리하고 (Information View), 정보가 특정 조건에 부합되면 이벤트를 발생한다(Event View). 이벤트가 발생되면 정보를 변경하거나 알고리즘을 실행한다(Active Object View).

4.1 Agent View

ALBM 클러스터 시스템의 Agent를 설계하기위한 기본 부분이다. 클러스터 시스템의 모든 구성 요소는 Agent 소프트웨어로 되어있고, 이 Agent를 설계하기 위해서 기본적으로 Agent View가 사용된다.

클러스터 시스템을 설계하기 위해서 설계자는 Cluster System Diagram으로 설계를 시작하게 된다. 클러스터 시스템은 다양한 Agent로 구성되고, 각 Agent는 무엇인가를 관리하는 Manager로 구성되어 있다. (그림 2)

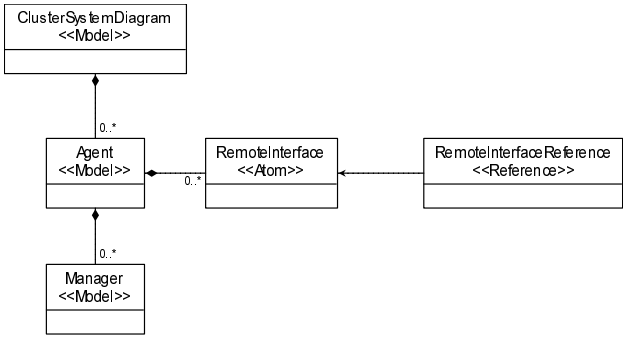

그림 2 - Agent View

Agent는 다른 Agent와의 상호작용이나 User Client을 위해서 Remote Interface를 가지고 있다. 즉, Remote Interface는 Agent가 다양한 상호작용을 지원하기 위해서 여러가지 형태로 동시에 가지고 있을 수 있다.

4.2 Information View

클러스터 시스템에서 사용되는 Agent의 기본적인 역할은 정보를 관리하는 것이다. 그림 3은 정보를 관리하기 위한 기본 구조를 보여준다.

Agent를 이루고 있는 많은 Manager들은 그 임무를 수행하기 위해서 기본적으로 정보들을 수집하고 관리한다. Information View에서는 이러한 기능을 제공하기 위한 구조를 제시한다. 이 구조는 정보를 저장하기 위한 Storage와 정보를 발생시키는 Source, 정보를 수집하는 Aggregator로 구성되어 있다.

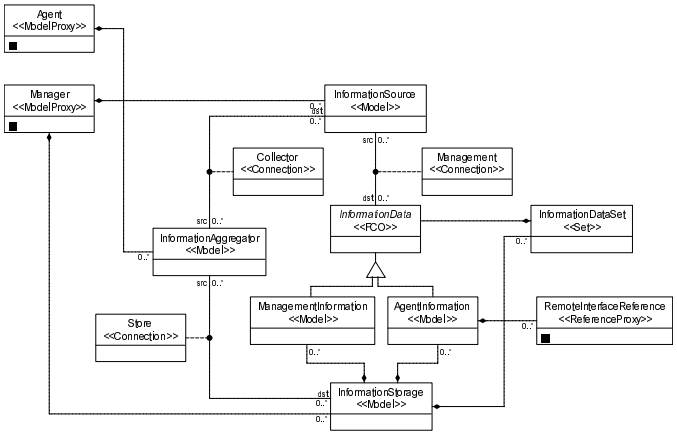

그림 3 - Information View

4.3 Event View

그림 4는 Event를 처리하기 위한 기본 구조를 보여준다. Event View에서 제시하는 구조는 Event는 보내는 Source와 비동기 전달을 지원하는 Broker, Event를 수신하는 Listener로 구성되어 있다. Manager와 Agent는 다양한 Event를 보내고 받게 된다.

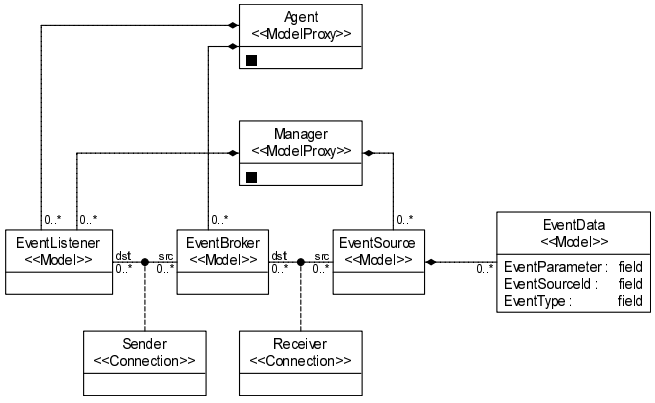

그림 4 - Event View

4.4 Active Object View

Event의 전달에 따라서 알고리즘을 실행하는 구조는

그림 5에서 볼수있다. Agent와 Manager는 수신되는 Event와 가지고 있는 Information에 따라서 알고리즘을 처리하게 된다. Active Object는 Event Listener 기능을 수행하면서 Algorithm을 Invoke한다.

그림 5에서는 알고리즘의 2가지 형태를 제시하고 있다. 알고리즘은 **Logic Algorithm**과 **Conditional Checking Algorithm**중 하나가 될수있다. Conditional Checking Algorithm은 상태의 검사 결과에 따라서 다른 LogicAlgorithm과 연동된다.

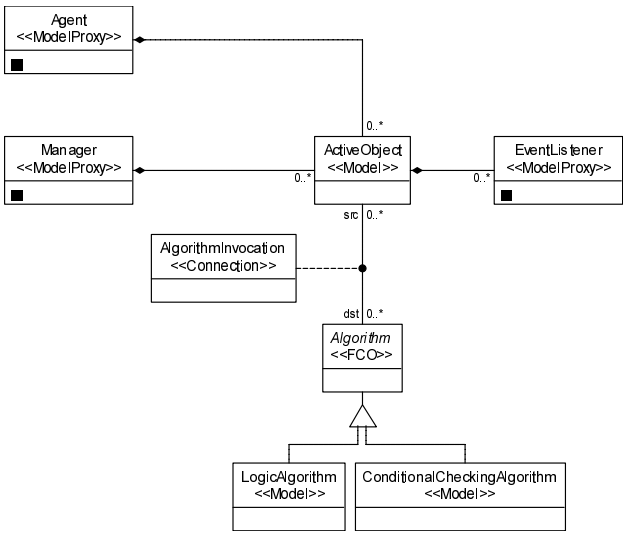

그림 5 - Active Object View

5. 결론

본 논문에서는 능동형 클러스터 관리 시스템을 위한 메타모델 설계를 제시하였다. 이 메타모델은 클러스터 관리 시스템의 중요한 기능인 **구성관리**와 **결함관리**를 고려하고 있다. 이 메타모델을 사용하면, 위의 특징을 가지는 에이전트 기반의 능동형 클러스터 관리 시스템을 쉽게 설계할 수 있다.

메타모델의 장점은 특정 분야에 대한 기반지식을 표현하여, 설계자로 하여금 구현 기반지식에 대한 깊은 이해가 없어도 전체 시스템을 쉽게 설계할 수 있도록 하는 것이다. 즉, 시스템을 개발할 때 실제 구현에 대하여 많은 부분을 **추상화**하여 설계자에게 필요한 부분만을 보여주게 된다.

다음 연구는 제시한 메타모델을 사용하여 클러스터 시스템을 설계하고, 이 모델을 실제 구현모델(Implementation Model)로 변환하는 **변환기(Transformation Interpreter)를 설계하고 구현**하는 것이다.

6. 참고문헌

[1] Stefanos Zachariadis, Cecilia Mascolo, Wolfgang Emmerich: “The SATIN Component System A Metamodel for Engineering Adaptable Mobile Systems,” IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL.32, NO.11, OCTOBER 2006

[2] Jin Liu, Keqing He, Bing Li, Chengwan He, Peng Liang: "A Transformation Definition Metamodel for Model Transformation," Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC’05)

[3] Yucong Duan, S.C. Cheung, XiaolanFu, Yuqing Gu: "A Metamodel Based Model Transformation Approach," Proceedings of the 2005 Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA’05)

[4] Junhua Wang, Soon-Kyeong Kim and David Carrington: "Verifying Metamodel Coverage of Model Transformations," Proceedings of the 2006 Australian Software Engineering Conference (ASWEC’06)

[5] Luis Pedro, Levi Lucio, Didier Buchs: "Principles for System Prototype and Verification using metamodel based Transformations," Proceedings of the Seventeenth IEEE International Workshop on Rapid System Prototyping (RSP'06)

[6] Institue for Software Integrated Systems, Vanderbilt University: “The Generic Modeling Environment,” http://www.isis.vanderbilt.edu/projects/gme/

[7] Object Management Group (OMG), MetaObject Facility, http://www.omg.org/mof/

[8] Esteban A. Pastor, R. T. Price: "Using Metamodels of Methodologies to determine the needs for reusability support," SSR ’97 MA, USA

[9] Máximo López S., Armando Alfonzo G., Joaquín Pérez O., Juan Gabriel González S., Azucena Montes R.: "A Metamodel to carry out Reverse Engineering of C++ Code into UML Sequence Diagrams," Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA'06)

[10] Alexandre Bragança, Ricardo J. Machado: "Extending UML 2.0 Metamodel for Complementary Usages of the <<extend>> Relationship within Use Case Variability Specification," 10th International Software Product Line Conference (SPLC'06)