제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

유연성 있는 이벤트 파싱 엔진의 설계 및 구현

윤태웅 ,민덕기
건국대학교 컴퓨터·정보통신공학과
E-mail : {taewoong, dkmin}@konkuk.ac.kr

Design and Implementation of Flexible Event Parsing Engine

Taewoong Yun, Dugki Min
Department of Computer Science and Engineering, Konkuk University

요 약

분산 시스템의 관리를 위해서 시스템 내의 각 컴포넌트에서 발생되는 많은 이벤트 중에 의미 있는 이벤트를 효과적으로 찾아내는 이벤트 파싱 엔진이 필요하다. 본 논문에서는 유연성 있는 이벤트 파싱 엔진을 제안한다. 특히 이벤트 파싱 엔진의 내부 구현을 위해서 기존의 룰 기반 접근방법을 변형한 새로운 접근방법을 제시한다. 본 엔진에서는 룰의 조건과 액션을 스크립트 객체화한 스크립트 객체 기반 접근방법과 룰에 해당하는 이벤트들을 찾아낼 때 보다 효과적인 검색을 지원하는 이벤트 토큰 기반 방식을 사용한다.

1. 서론

수많은 서버들로 이루어진 분산 시스템을 관리하는 일은 무척 어려운 일이다. 시스템을 관리하기 위해서는 먼저 현 시스템에 대한 분석이 필요하며 정확한 분석을 위해서는 시스템 성능 분석을 해야 한다. 시스템 성능 분석을 위해서는 시스템에서 발생하는 성능 정보를 모니터링 하는 부분이 필요하다. 모니터링 보다 더 근본적으로 되어야 하는 것은 모니터링을 통한 성능 정보가 발생한 시점에서의 시스템의 상태를 알아내는 것이 필요하다. 즉 시스템을 이루는 각 서버(시스템을 이루는 각각의 컴포넌트)에서 발생하는 이벤트를 분석할 필요가 있다[1]. 이벤트 분석이 없는 시스템은 모니터링을 통한 성능정보만 얻을 수 있고 얻어진 성능정보의 원인을 알 수가 없다. 그러나 이벤트를 분석 할 수 있는 시스템은 이벤트 발생 상황에 따른 정보의 차이와 원인을 분석할 수 있게 될 뿐만 아니라 발생하는 이벤트를 통한 Proactive Management[7]가 가능하게 된다. 이러한 이유로 시스템 각 컴포넌트에서 발생하는 다양한 이벤트 파싱을 위한 유연한 이벤트 파싱 엔진이 필요하다.

본 연구에서는 유연한 이벤트 파싱 엔진을 위한 접근방법으로 기존의 룰 기반의 접근 방법을 변형한 새로운 접근방법을 제시한다. 룰의 조건과 액션을 스크립트 객체화한 스크립트 객체 기반 접근방법과 룰에 해당하는 이벤트를 찾아낼 때 보다 효과적인 검색을 지원하는 이벤트 토큰 기반 접근방법을 사용한다.

본 논문의 구성은 2장에서 이벤트 시스템의 3계층의 구조와 역할에 대해서 설명하고, 3장에서는 기존의 이벤트 파싱 접근방법의 장단점을 비교 분석한다. 4장에서는 논문에서 제시하는 접근방법에 대해서 설명하고 5장에서는 4장에서 제시하는 접근방법을 사용한 이벤트 파싱 엔진의 구조에 대해서 설명한다. 마지막 장에서는 결론과 향후 추가되어야 할 사항으로 마무리한다.

2. 이벤트 시스템 구조

이벤트 기반의 시스템이란 시스템의 각 컴포넌트에서 특정한 상황에 발생하는 이벤트가 미리 정의되어 있고 컴포넌트와 컴포넌트들이 이벤트를 주고받는 시스템을 말한다. 즉 서로 연관되어 있는 컴포넌트들은 상대편의 컴포넌트로부터 들어온 이벤트와 자신의 상태를 비교해서 작동하는 것이다.

이러한 이벤트와 비슷한 개념으로는 메시지가 있는데, 메시지는 시스템에서 필요로 하는 명령, 이벤트, 정보 등을 포괄하는 개념이고 이벤트는 전달되는 메시지 중 특별한 의미를 가지고 해석할 수 있는 메시지를 말한다.

이벤트 시스템의 구조는 그림1 과 같이 세 개의 계층으로 나뉘어 지며 Message Communication은 네트워크를 통해서 메시지를 보내는 계층이고 Notification Server, Channel Server는 각 노드에서 발생하는 이벤트를 처리하고 다른 노드로 이벤트를 전달하는 계층이다. Notification Server를 통해서 서로 떨어져 있는 컴포넌트들의 Direct 통신이 가능하며 서로 직접적으로 연결될 수 없는 컴포넌트들은 Channel Server를 통해서 간접적으로 Indirect통신을 하게 된다. 이벤트 파싱을 위한 룰 기반의 엔진은 Rule Engine, Correlation 계층에 존재한다. Rule Engine, Correlation 계층은 하나의 노드 혹은 하나의 컴포넌트, 또는 전체 시스템에서의 이벤트 발생 원인 및 관계 분석과 분석된 결과를 다시 시스템에 적용시키는 역할을 담당한다.

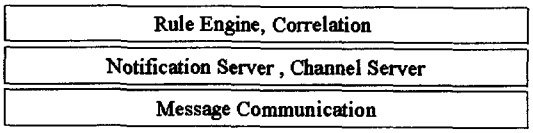

그림 1 이벤트 시스템의 구조

3. 관련연구

3.1. Casual-based 모델 접근방법

Casual-based 모델 시스템[3]은 시스템에서 일어 날 수 있는 다양한 이벤트와 원인을 저장하고 있다가 어떤 이벤트가 발생 시 발생한 이벤트의 원인을 찾아내는 모델이다. 구성은 세 개의 부분으로 되어 있으며 Dynamic Fact Repository, Event Knowledge Model, 그리고 Correlator로 구성된다. Dynamic Fact Repository는 네트워크나 시스템의 정보를 저장하고 있고, Event Knowledge Model은 시스템에서 일어날 수 있는 다양한 이벤트와 원인을 저장하고 있다. 이러한 이벤트와 원인과의 인과 관계는 트리, 그래프, 룰, 유한 상태머신 등으로 표현 할 수 있다. 마지막으로 Correlator는 Knowledge Model을 가지고 시스템 모니터로부터 들어온 이벤트를 통해서 시스템으로부터 발생한 이벤트의 원인과 발생 장소를 분석해 낸다. 다른 접근방법 중에 일반적이며, 추상적인 접근방법이다. 실제 구현으로 위해서는 3.2,절 및 3.3절에서와 같은 구체적인 접근방법을 사용해야 한다.

3.2. 코드 기반 접근방법

코드 기반 접근방법[3]은 CASUAL MODEL 접근방법의 변형으로 발생할 수 있는 이벤트를 코드(Code)로 미리 정의 한 후 발생하는 이벤트에 따라서 미리 정의된 코드(Code)표에 있으면 적절한 액션을 취하는 접근방법이다.

장점으로는 코드 값만 비교하므로 알고리즘이 단순하고 실행 속도 또한 빠르다. 단점으로는 다양한 환경에서 일어나는 모든 이벤트를 코드 화 할 수 없기 때문에 사용하기에는 비현실적이다 는 단점이 있다. 또한 다양한 이벤트 파싱을 위한 접근방법으로는 부적합하다.

3.3. 룰 기반 접근방법

룰 기반의 접근방법[2,3]은 이벤트 파싱의 방법으로 룰을 정의하고 정의된 룰의 조건에 해당하는 이벤트를 검색해서 룰을 적용하는 방법이다. 구조는 세 개의 컴포넌트로 이루어진다. 각 각 User-Interface, Inference Engine, Knowledge(or Rule) Base 이며 도메인 지식을 룰로 표현 가능한 전문가 시스템의 한 타입이다. 룰은 "IF condition THEN action"의 형태를 가진다[8]. 조건에 맞는 이벤트가 발생했을 때 추론 엔진이 조건을 판별하고 액션을 취하는 방식이다. 코드 기반의 방식처럼 이벤트를 하나의 코드로 표시하는 것 이 아니다. 이벤트의 나열(발생순서)을 표시함으로써 이벤트의 어떠한 흐름을 판별해서 알아내는 것이다.

장점으로는 코드 기반 접근방법 보다는 다양한 형태의 이벤트와 룰을 정의할 수 있다는 점이 있다. 단점으로는 룰에 적용되는지를 판별하는 과정이 비교적 복잡하고, 이벤트의 흐름과 조건을 다 확인해야 하기 때문에 코드 기반 접근방법보다 느리다. 또한 조건과 액션은 스크립트 같은 언어로 정의되어 있으며, 따로 조건과 액션을 정의하는 언어를 알아야 하고 그 기능이 다양하지 못해서 복잡한 조건이나 다양한 액션이 불가능하다.

4. 이벤트 파싱 엔진 접근방법

도입부에 작성한 접근방법에 대해서 본 장에서는 접

근 방법과 이벤트 파싱 엔진의 구현에 있어서의 특징인 스크립트 객체 기반 접근방법, 이벤트 토큰 기반 접근방법과 이벤트 파싱 엔진의 확장성에 대하여 살펴보겠다.

4.1. 스크립트 객체 기반

스크립트 객체 기반 접근방법은 룰 기반 접근방법의 변형으로 룰을 사용해서 이벤트를 파싱 하는 방법이다. 하지만 룰의 조건과 액션을 정의하는데 제한된 문법을 가지는 스크립트 언어로 작성하는 대신에 스크립트 역할을 하는 후크(Hook) 객체[5]를 정의해서 사용한다. 구현에 있어서 스크립트 객체를 만들어 내기 위해서 자바 언어를 선택하였으며, 자바의 객체 지향적인 측면과 논문에서 구현하는 이벤트 파싱 엔진이 자바로 개발되는 이유(이벤트 파싱 엔진 자체에도 영향을 가할 수 있는 구조)도 포함되었다.

스크립트 객체 기반의 장점으로는 조건과 액션을 정의할 스크립트 언어를 따로 정의 할 필요가 없고, 조건과 액션코드가 인터프러터 방식이 아닌 자바 바이트 코드로 미리 컴파일되어 있는 컴파일러 방식으로 작동되기 때문에 속도가 빠르다는 장점이 있다. 더욱이 자바언어 내부에 정의되어있는 라이브러리를 사용 할 수 있기 때문에 보다 복잡한 조건문과 액션을 처리 할 수 있게 된다. 즉, 다양한 문자열 처리, 숫자 처리, 논리 처리가 가능해 지는 것이다. 또 하나의 장점으로는 실행 가능한 액션코드로 기존의 시스템(자바)에 쉽게 통합할 수 있기 때문에 이벤트를 적용할 시스템에 보다 많고, 강력한 액션을 취할 수 있게 된다는 장점이 있다.

4.2. 이벤트 토큰 기반

이벤트 토큰 기반 방식이란 이벤트 흐름에 맞는 룰을 검색하는 과정에서 룰 안에 정의된 이벤트의 리스트와 조건을 전부를 검색하는 것이 아니라 룰에 적용 될 수 있는 이벤트 토큰 리스트(도메인과 이벤트 타입) 즉, 이벤트 토큰을 사용함으로써 룰에 적용되어야 하는 룰을 보다 빠르게 검색하는 방식이다. 이 방법은 일반적인 컴파일러가 문자열에서 의미 있는 토큰(Token)을 찾아내는 방법[6]과 동일하게 처리 할 수 있다. 또한 언어를 표현하는 정규식에 있는 연산자 기호(*, ., [])등을 표현 할 수 있어서 더욱 실제 환경에서 발생 할 수 있는 이벤트를 비슷하게 묘사 할 수 있다는 장점이 있다.

4.3. 이벤트 파싱 엔진의 확장성

본 논문에서 제시하는 이벤트 파싱 엔진의 특징 중 하나는 룰 기반의 이벤트 파싱 엔진의 일반화 모델을 제시하는데 있다. 입력 부분에 있어서는 코바에 있는 StructuredEvent 인터페이스[4]를 사용함으로써 다른 이벤트도 파싱 가능하게 된다. 또한 Event Correlation을 위한 기본적인 분석을 해주거나 Security Policy 적용을 위한 Policy분석을 위해서도 StructuredEvent 인터페이스만 따른다면 분석 가능하게 된다. 시스템 내에서 알고리즘을 어떠한 상황에 변경해서 적용해야 하는 경우나 Policy-Based Management에도 사용이 가능하게 이벤트 파싱 엔진을 일반화하는 구조로 구현하였다.

5. 이벤트 파싱 엔진의 구조

이벤트 파싱 엔진의 구조는 그림 2와 같이 이루어져 있으며 크게 세개의 패키지로 구성된다. Information Package는 이벤트 파싱 엔진의 룰 정보, 엔진의 정보를 관리하고, Engine Package는 룰 엔진의 핵심 골격을 담당하고, 엔진을 시작, 중지, 하는 역할을 한다. 마지막으로 Parser Package는 Information Package에서 정의된 룰을 가지고 파싱 테이블을 구성한 뒤 실시간으로 발생하는 이벤트를 가지고 적용할 수 있는 룰을 찾는 작업을 한다.

5.1. Information Package

RuleInfo는 하나 이상의 룰을 정의한다. 룰 정의는 룰 이름, 룰이 두 개 이상 발견 시 적용되는 순서를 정하는 우선순위, "도메인 이름:이벤트 타입이름"을 나타내는 이벤트 토큰 이름, 자바 문장으로 이루어진 조건 코드, 자바 문장으로 이루어진 액션 코드로 이루어진다. 또한 이벤트 파싱에 필요한 룰 정의를 XML파일에 저장해서 이벤트 파싱 엔진을 사용하려는 곳에서 룰 정보를 삽입, 삭제, 수정함으로써 발생하는 이벤트 따른 액션이 달라지게 할 수 있게 하였다.

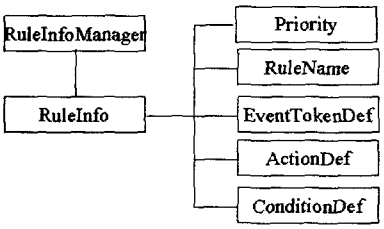

그림 2 Information package 구조

5.2. Engine Package

룰 엔진은 인터페이스를 상속함으로써 알고리즘을 변경 할 수 있게 한다. 룰 엔진은 그림 3와 같이 세

개의 컴포넌트를 갖는다. EventBufferManager는 실시간으로 들어온 이벤트를 관리한다. 특정 시간이 지날 경우 오래된 이벤트는 삭제하는 일을 담당하고, RuleInfoManager는 룰 관련 정보를 관리하며, JavaCodeBuilder는 엔진 최초 가동 시 조건과 액션을 자바 코드로 정의된 부분을 실행가능 한 인터페이스 상속 객체로 변경한다. 엔진 실행 시 조건을 비교하거나 액션을 실행 할 때 JavaCodeBulider를 통해서 생성된 비교가능 하거나 실행 가능한 객체를 사용하게 된다.

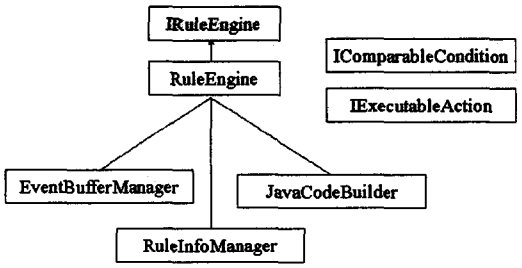

그림 3 룰 파싱 엔진 구조

5.2.1. IComparableCondition Interface

IComparableCondition은 룰의 조건 스크립트 문장을 비교 가능한 객체로 변환 시 상속받는 인터페이스이다. 파라미터 인 Events는 룰에 정의 된 EventTokenList를 의미한다. 인터페이스 안에 정의된 Compare함수에서 파라미터인 Events를 가지고 이벤트의 실제 데이터를 사용해서 조건을 만들어 낸다. 즉, 발생된 이벤트의 데이터를 확인하거나 이벤트의 머리말 정보도 활용 하게된다. 이러한 인터페이스를 상속받은 객체는 JavaCodeBuilder에 의해서 룰 정의 정보에서 실시간에 비교해 볼 수 있는 객체로 변환되어 진다.

5.2.2. IExecutableAction Interface

IExecutableAction 인터페이스는 룰의 액션 스크립트 문장을 실행 가능한 객체로 변환 시 상속 받는 인터페이스이다. 룰에 의해서 적용되어야 하는 액션 코드들이 인터페이스의 Execute함수 안에 들어가게 된다. 내부 함수 호출은 물론 자바에 정의된 함수 호출도 가능하고 원격메서드호출(RMI)나 소켓을 사용하면 외부에 있는 시스템이나 서버에 영향을 줄 수도 있다. 이 인터페이스를 상속받은 객체는 JavaCodeBuilder에 의해서 실시간에 실행 가능한 액션 객체로 변환되어진다.

6. 결론

본 논문에서는 시스템에 이벤트가 왜 필요한지, 이벤트 기반의 시스템은 어떻게 이루어져야 하는지에 대해서 설명하였고 그러한 이벤트 기반의 시스템이 필요로 하는 이벤트 파싱 엔진의 접근방법과 구조, 설계상의 이슈에 대해서 살펴보았다. 또한 논문에서 제시하는 스크립트 객체 기반, 이벤트 객체 기반 이벤트 파싱 엔진을 위한 설계상의 이슈도 살펴보았다.

참고문헌

[1] Yongseok Park, "Event correlation " ,IEEE Potentials , Volume: 20 Issue: 2 , Apr/May 2001 ,Page(s): 34 -35.

[2] Appleby, K., Goldszmidt, G., Steinder, M., "Yemanja - a layered event correlation engine for multi-domain server farms ",Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on , 2001 ,Page(s): 329 -344.

[3] Robert D. Gardner & David A. Harle," Methods and Systems for Alarm Correlation " ,IEEE, 1996.

[4] CORBA Notification Specification 1.0.1, August 2002

[5] Mark Grand, "Patterns in Java Volume 1 : A Catalog of Reuseable Design Patterns Illustrated with UML", wiley computer publishing, 1998.

[6] Elliot Berk, "JLex: A Lexical Analyzer Generator for Java(TM)" available at URL:
http://www.cs.princeton.edu/~appel/modern/java/JLex/

[7] Yoonhee Kim, Hariri, S., Djunaedi, M., "Experimental results and evaluation of the proactive application management system(PAMS)", Conference Proceeding of the IEEE International , Feb 2000, Page(s): 76 -82.

[8] Jakobson, G., Weissman, M., "Alarm correlation ", IEEE Network , Volume: 7 Issue: 6 , Nov 1993 , Page(s): 52 -59.