클러스터 환경에서 Direct 이벤트 통지 서비스의 설계 및 구현

염귀덕, 김석동, 민덕기
건국대학교 컴퓨터 · 정보통신공학과
E-mail:{kdyeom, jung945, dkmin}@konkuk.ac.kr

Implementation of Direct Event Notification Service on Cluster Environment

Kiduk Yeom, Suk-Dong Kim, Dugki Min
Dept of Computer Science and Engineering, Konkuk University

요 약

가격대 성능비가 우수한 클러스터 시스템의 보급이 확산되어 많이 사용되고 있다. 클러스터 시스템을 구성하는 컴포넌트들은 다양한 이벤트를 발생시키고 통지하는 상호작용을 하기 때문에 시스템을 효과적으로 관리하기 위해서는 이벤트를 직접 또는 간접적으로 통지해 주는 이벤트 통지 서비스가 필요하다. 본 논문은 클러스터 환경에서 컴포넌트들 간에 발생하는 이벤트를 직접적으로 통지하는 이벤트 통지 서비스를 제안한다. 이 이벤트 통지 서비스는 컴포넌트들 간에 발생한 이벤트를 직접적으로 통지하기 때문에 전송속도가 빠르며, 통지 시 필요에 따라 다양한 이벤트 핸들링 모듈을 적용할 수 있는 유연한 구조로 설계되었다. 또한, 신뢰성 있는 통신을 위하여 UDP를 확장한 하부 데이터 통신 인프라를 사용하였다.

1. 서론

인터넷 사용의 급속한 확대와 다양한 종류의 서비스로 인하여 대용량의 부하를 처리해내는 가격대 성능비가 우수한 클러스터 시스템의 사용이 점차 확산되고 있다[1]. 클러스터 시스템을 구성하는 컴포넌트들은 다양한 이벤트를 발생시키고 통지하는 상호작용을 하기 때문에 이벤트 통지 서비스는 필수적이다. 일반적으로, 시스템의 컴포넌트들 간에 이벤트를 통지하는 방식에는 두 가지가 있다. 첫 번째는 이벤트를 발생시키는 생산자(Supplier)와 이벤트를 통지받는 소비자(Consumer)가 채널과 같은 중간의 중재자를 통해서 통신하는 간접적인 통신(Indirect Communication)이다[2]. 이 간접통신 방식은 공급자와 생산자간 결합력이 낮은 장점이 있지만 통지할 이벤트가 많거나 빠른 통지가 필요한 긴급한 이벤트인 경우에 전송이 지연되는 단점이 있다. 두 번째는 생산자와 소비자를 직접 연결하는 직접적인 통신(Direct Communication)이다. 빠른 실시간 통지가 필요한 경우에 필요한 방식이다. 본 연구에서 고려하는 이벤트 통지 서비스는 생산자와 소비자가 직접적으로 통신하는 방식이다.

본 논문은 클러스터 환경에서 컴포넌트들 간에 발생하는 이벤트를 직접적으로 통지하는 이벤트 통지 서비스를 제안한다. 이 이벤트 통지 서비스는 이벤트 생산자와 소비자가 직접적으로 통신하기 때문에 전송속도가 빠르며, 통지 시 필요에 따라 다양한 이벤트 핸들링 모듈을 적용할 수 있는 유연한 구조로 설계되었다. 또한 신뢰성 있는 통신을 위하여 UDP를 확장한 하부 데이터 통신 인프라를 사용하였다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구를 살펴보고, 3장에서는 클러스터 환경에서 Direct

이벤트 통지 서비스의 필요성과 제안한 Direct 이벤트 통지 서비스의 구조를 제시한다. 4장에서는 이벤트 통지 서비스의 설계 및 구현을 설명하고 신뢰성 있는 통신을 위하여 UDP를 확장한 하부 데이터 통신 인프라 구조에 관해서 살펴본다. 마지막 장에서는 결론과 향후 추가되어야 할 내용으로 마무리한다.

2. 관련 연구

기존의 이벤트 통지 서비스에 대하여는 많은 연구들이 이루어져 왔다[3, 4, 5]. 본 논문에서는 그 중에서 가장 보편적으로 알려지고 많이 사용하는 OMG(Object Management Group)에서 제시하고 개발한 CORBA(Corba Object Request Broker Architecture)의 Event Service, Notification Service와 Sun Microsystems의 JMS(Java Message System) 그리고 콜로라도 대학의 SERL(Software Engineering Research Laboratory) 연구실에서 개발한 SIENA(Scalable Internet Event Notification Architectures)에 관해서 설명한다.

CORBA의 이벤트 통지 서비스는 생산자(Supplier)와 소비자(Consumer)가 중앙집중적인(Centralized) 채널을 통해서 이벤트를 주고받는 채널기반의 이벤트 통지 서비스이다. 채널을 통한 간접 통신을 하므로 이벤트를 발생하는 생산자(Supplier)와 이벤트를 통지받는 소비자(Consumer) 간 결합력이 낮다는 장점을 지닌다. 이벤트 소비자는 관심있는 이벤트에 대한 통지를 채널에다 등록시킬 수 있으며 자신이 어떤 통신 모드로 동작할지를 지정할 수 있다. 생산자와 소비자는 서로에 대한 정보 없이 이벤트를 전달할 수 있기 때문에 새로운 생산자나 소비자를 쉽게 시스템에 추가할 수 있는 확장성있는 구조를 가진다. 주로 전자통신 영역에서 사용하도록 개발되었다.

자바 메시지 서비스(JMS)는 Sun Microsystems에서 만든 기업환경의 분산 어플리케이션의 소프트웨어 컴포넌트들 사이의 메시지 전달이 가능하도록 개발되었다. 컴포넌트들 사이의 신뢰성있고, 확장성 있는 구조에 중점을 두었다는 점에서 우리의 구조와 비슷하지만 JMS가 제공하는 기능을 이용하기 위해서는 JMS Provider를 구현한 메시징 제품을 이용한다.

SIENA는 계층적인 중앙집중식 채널(Hierarchical Centralized Channel)방식과 지점간 연결(Peer-To-Peer) 방식을 둘 다 지원하는 WAN(Wide Area Network) 환경의 이벤트 통지 서비스이며 이벤트를 통지받을 때 Content 단위로 통지받는 Content기반 이벤트 통지 서비스이다.

3. 클러스터 환경에서 Direct 이벤트 통지 서비스

클러스터 시스템은 외부적으로는 단일 시스템으로 작동하지만 내부적으로는 여러 노드들이 한데 묶여서 정보를 공유한다. 이러한 시스템에서 이벤트 통지 서비스를 사용하기 위해서는 다음과 같은 기능들이 보장되어야 한다. 첫째, 이벤트 통지 시 전송속도가 빨라야 한다. 시스템의 각 노드에서는 다양한 이벤트가 발생하는데 이벤트 마다 우선순위(Priority)가 있어서 특히, 우선순위가 높은 혹은 긴급히 통지해야 되는 이벤트인 경우는 빠른 전송속도로 통지되어야 한다. 둘째, 유연성(Flexibility)이다. 이벤트를 통지할 목적지 노드가 죽은 경우, 설정을 바꾸어서 다른 노드로 이벤트가 전달될 수 있도록 실시간으로 송수신 이벤트 전달 흐름을 변경할 수 있어야 한다. 마지막으로, 다양한 이벤트 핸들링(Event Handling)이다. 이벤트 핸들링은 이벤트 필터링(Event Filtering), 이벤트 로깅(Event Logging), 그리고 이벤트 암호화 및 복호화(Event Encryption and Decryption)를 말한다.

이벤트 필터링은 전달할 이벤트와 전달해서는 안되는 이벤트를 걸러 주는 기능을 말한다. 이벤트 로깅은 저장이 필요한 중요한 이벤트를 이벤트 저장소에 저장하는 기능이며 특히, 보안이 요구되는 이벤트인 경우는 암호화 혹은 복호화 기능을 적용시켜서 보안을 유지할 수 있다. 이러한 다양한 이벤트 핸들링 모듈은 필요에 따라서 사용하므로 Hook 객체로 구현해서 플러그 인 앤 플레이(Plug-in and Play)로 작동하면 시스템의 성능에 영향을 미치지 않으면서 효율적으로 사용할 수 있다.

그림 1은 본 논문에서 제시한 Direct 이벤트 통지 서비스를 클러스터 환경에 적용한 구조를 나타낸다. 클러스터 시스템의 각 노드들은 자바가상머신(JVM) 에 이벤트 통지 서비스(Event Notification Service)를 탑재하고 있다. 서비스 초기화 단계에서, 이벤트 소비자들은 관심있는 이벤트에 대한 통지를 리스너 (Listener)형태로 서비스에 등록할 수 있으며 Notification Server는 목적지 노드의 정보를 읽어와서 실행한다. 이벤트 통지 서비스는 크게 다음과 같은 컴포넌트들 즉, 메인 서버인 Notification Server,

다양한 이벤트 핸들링 모듈을 제공하는 Hook 객체로 작동하는 Event Handler, 이벤트 통지시 DisseminationInfo 정보를 가져오는 Event Dispatcher, 그리고 다양한 이벤트를 StructuredEvent 타입으로 생성하는 Event Delegator로 구성된다.

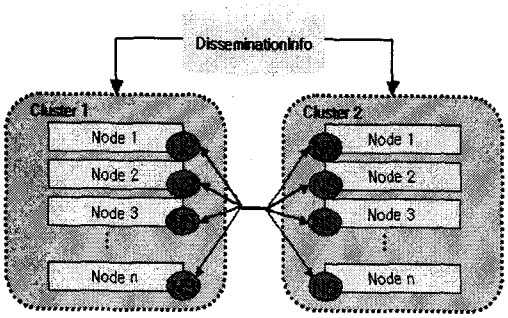

그림 1 클러스터 환경에서 Direct 이벤트 통지 서비스의 구조

각 노드에서 발생한 이벤트를 통지할 때는 DisseminationInfo라는 라우팅 테이블에서 목적지 노드의 정보를 가져와서 직접적으로 통지하게 된다. DisseminationInfo는 XML 파일 형태로 이벤트 타입(Event Type)당 목적지 정보로 구성되며 클러스터 내 모든 노드들이 공유하는 정보이다. DisseminationInfo 내의 모든 정보는 Dissemination Manager가 설정하며 관리한다. 만약, 이벤트 통지시 목적지 노드가 죽은 경우는 DisseminationInfo의 설정만 바꾸면 Notification Server가 바뀐 설정 정보를 읽어와서 다른 목적지 노드로 이벤트를 전달할 수 있는 유연한 구조로 가지고 있다.

4. 이벤트 통지 서비스

본 장에서는 이벤트 통지 서비스의 컴포넌트들에 대하여 자세히 살펴보고 신뢰성 있는 통신을 위한 UDP를 확장한 하부 데이터 통신 인프라 구조에 관해서 설명한다.

4.1 이벤트 통지 서비스의 설계 및 구현

그림 2와 같이 이벤트 통지 서비스는 크게 Notification Server, EventHandler, Event Dispatcher, Event Logger, 그리고 Event Supplier 및 Consumer 응용프로그램(application)과 연동하기 위한 Event Delegator로 구성되어 있다. 여기서는 이벤트의 발생 및 통지까지의 전송과정을 살펴봄으로써 각 컴포넌트가 하는 역할을 자세히 알아본다.

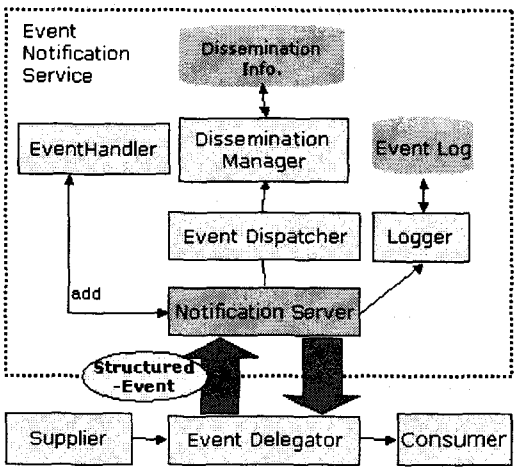

그림 2 Structure of Event Notification Service

이벤트를 발생시키려는 생산자(Supplier) 어플리케이션들은 Event Delegator를 통해서 다양한 이벤트를 발생시킨다. Event Delegator는 발생된 이벤트를 표준 이벤트로 생성해 내는 일종의 이벤트 생성기로써 생산자당 혹은 소비자당 하나로 할당된다. 본 논문에서 구현한 이벤트 형식은 CORBA의 Notification Service에서 정의한 StructuredEvent 타입을 사용하고 있다[6]. StructuredEvent 타입에는 도메인 이름, 이벤트 타입 이름 그리고 이벤트 이름으로 구성되는 헤더 필드(Header Filed)가 있어서 이벤트에 대한 자세한 정보를 알 수 있으며 CORBA의 표준을 따름으로써 이기종의 이벤트 통지 서비스와 연동을 용이하다. 생성된 StructuredEvent 타입은 XML 형태로 변환한 후 통신 하부 구조를 통해서 Notification 서버로 전달된다. XML 형태로 전송된 이벤트를 수신한 Notification 서버는 메인 서버로써 내부에 Event Handler를 가지고 있어서 이벤트 처리에 관련된 다양한 이벤트 핸들링 모듈을 필요에 따라서 실행할 수 있다. Event Handler는 이벤트 필터링, 이벤트 로깅, 그리고 이벤트 암호화/복호화 기능으로 구성되며 Hook 객체로 구현되어서 플러그인 앤 플레이(Plug-in and Play)로 작동된다. 이러한 Hook 객체는 필요한 경우에만 실행시키므로 시스템의 성능에 영향을 미치지 않고 효율적으로 사용할 수 있다는 장점을 가진다. Notification 서버로부터 이벤트를 전달받은 Event Dispatcher는 이벤트를 분배하는 역할을 한다. 즉, DisseminationInfo로부터

이벤트가 통지될 목적지인 Target 정보를 가져와서 XML 타입을 StructuredEvent 타입으로 변환한 후 통신 하부를 통해서 Event Delegator에 전달하게 된다. Event Delegator에는 서비스 초기화 단계에 관심있는 이벤트의 통지를 리스너(Listener) 형태로 등록한 이벤트 소비자(Consumer) 어플리케이션들이 있어서 관심있는 이벤트를 통지받게 된다. DisseminationInfo는 이벤트 타입당 Target정보로 구성되며 DisseminationManager는 Target 정보를 설정하며 로딩이나 저장과 같은 정보관리 로직을 실행하는 관리자 역할을 한다.

4.2 신뢰성 있는 통신 하부구조

이벤트 통지 서비스는 빠른 전송 속도와 함께 전송에 대한 Quality가 보장이 되어야 한다. 데이터 전송 방식은 신뢰성있는(Connection-Oriented) TCP기반의 방식과 비신뢰적인(Connectionless) UDP 기반의 전송방식이 있다. TCP기반의 전송방식을 사용하는 경우에는 UDP기반의 전송방식보다 속도가 느리며, 통신상의 문제가 발생한 경우(예를 들어, 일부 네트워크 연결이 일시적으로 단절되는 경우) 정상적인 시스템에 데이터 전송을 할 때도 영향을 미치게 되어 전체시스템을 멈추게 하는 문제점을 발생시킬 수 있다. 이러한 문제점은 UDP기반의 전송방식을 사용하는 Message Communication을 통신 하부구조로 사용함으로써 해결될 수 있다. 본 논문에서는 UDP를 확장한 하부 데이터 통신 인프라를 설계하였으며 현재 구현 중이다. 그림3은 UDP를 확장한 Message Communicaiton의 구조를 나타낸다.

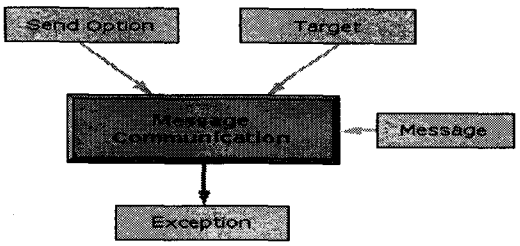

그림 3 Strucuture of Message Communication

신뢰성있는 메시지 통신을 위해서 송신자가 메시지를 보낼 때는 목적지(Target)에 대한 정보와 송신 옵션(Send Option)인 전송 간격(Interval)과 카운터(Counter)를 두어서 수신자로부터 응답이 오면 올바르게 송신이 된 것이고 만약, 응답이 오지 않을 경우는 전송 간격만큼 대기하거나 카운터 수만큼 반복해서 메시지가 신뢰성 있게 도착하도록 하였다. 메시지 전송 과정에서 발생하는 예외는 Exception클래스에서 사용자가 정의한대로 발생시키게 된다.

5. 결론 및 향후 연구

본 논문에서는 클러스터 환경에서 컴포넌트들 간에 발생하는 이벤트를 직접적으로 통지하는 이벤트 통지 서비스를 제안했다. 이 이벤트 통지 서비스는 이벤트 생산자와 소비자가 직접적으로 통신하기 때문에 전송속도가 빠르며, 통지 시 필요에 따라 다양한 이벤트 핸들링 모듈을 적용할 수 있는 유연한 구조로 설계되었다. 또한 신뢰성 있는 통신을 위하여 UDP를 확장한 하부 데이터 통신 인프라를 사용하였다. 향후 이벤트 통지 서비스의 성능 측정과 평가에 관한 연구가 필요하다.

6. 참고 문헌

[1] Rajkumar Buyya, "High Performance Cluster Computing vol.1", Prentice Hall, 1999

[2] M Molowidzki, "Advanced Event Filtering Approach For CORBA-Based Management Systems", IEEE 2000

[3] Object Mangement Group, "The Common Objecct Request Broker: Architecture and Specification, Revision 2.2

[4] Sun Microsystems, "JMX(Java Management Extension)//JDMK(Java Dynamic Managemement Toolkit)" http://java.sun.com/products

[5] Antonio Carzaniga, David S. Rosenblum, Alexander L. Wolf "Design and Evaluation of a Wide-Area Event Notification Service" ACM Transactions on Computer Systems, Vol. 19, No. 3, August 2001, Pages 332-383

[6] Srinivasan Ramani, BalaKrishnan Dasarathy, and Kishor S. Trivedi "Reliable Messaging Using the CORBA Notfication Service" IEEE 2001

[7] M TOMONO, "An Event Notification framework based on Java and CORBA" IEEE, 2000