EJB(Enterprise JavaBeans) 컨테이너 서버의 클러스터링에 관한 연구

김성훈°, 정승욱, 서범수, 김중배
한국전자통신연구원 모바일응용서버연구팀
{saint°, swjung, bsseo, jjkim}@etri.re.kr

A Study on Clustered EJB Container Server System

Sung-Hoon Kim°, Seoung-Woog Jung, Bum-Soo Seo, Joong-Bae Kim
Mobile Application Server Research Team
Electronics and Telecommunications Research Institute

요 약

EJB(Enterprise JavaBeans)는 Sun사에서 추진하는 분산 트랜잭션 기반의 엔터프라이즈 어플리케이션을 위한 컴포넌트 컴퓨팅 아키텍처이다. 또한 J2EE 표준 플랫폼에서 서버 측 컴포넌트를 위한 아키텍처로써, 다중 플랫폼 또는 다중 서버에 이식 가능한 비즈니스 로직을 표현하는 코어 기술이다. 본 논문에서는 이러한 EJB 아키텍처에 따라 구현된 EJB 서버 시스템을 클러스터링 환경으로 확장하여 고가용성 및 확장성을 얻고자 할 때 고려해야 할 사항들과 시스템 구현에 따른 구체적인 설계 방안을 제시한다.

1. 서 론

최근 J2EE(Jave2 Enterprise Edition) 기반의 웹 응용서버는 외국의 주요 서버 벤더들이 운영체제에 기본으로 탑재하여 배포할 만큼 인터넷을 기반으로 하는 e-비즈니스 분야에서 그 중요성이 날로 커지고 있다.

EJB(*Enterprise JavaBeans*)는 기업형 응용 시스템 개발자들의 요구 사항을 충족하기 위하여 SUN사에 의해 추진되는 산업계 표준의 분산 컴포넌트 아키텍처이다. 또한 EJB는 기업형 비즈니스 응용을 작성하는 개발자들에게 빈 객체의 생명주기 관리, 인증 및 접근 권한 검사, 분산 트랜잭션 처리, 사용자 및 객체의 상태관리, 멀티 쓰레드, 컨넥션 풀링, 영속성 관리 등과 같은 복잡한 하부 미들웨어의 코딩없이 손쉽게 개발할 수 있도록 함으로써 개발 생산성과 비즈니스 업무의 호환성, 로직의 재사용성을 높일 수 있는 산업계 표준이다[1].

실제로 CBD(Component Based Development)가 S/W분야에서 주된 개발 방법론으로 자리를 잡아감에 따라 EJB 기반의 응용 개발이 더욱 확산되고 있으며, 개발 방법론 측면에서뿐만 아니라 응용서버의 용도도 금융등과 같은 업무 핵심적인 분야에서도 EJB 기반의 응용서버의 중요성이 부각되고 있는 추세이다.

그러나 SUN사의 EJB 표준은 분산 객체 컴포넌트 기반의 개발 방법론에 관한 표준은 기술하고 있지만 이를 지원할 서버의 구현 방법론에 대해서는 기술하고 있지 않다. 따라서 EJB를 지원하는 모든 웹 응용서버 벤더들은 자체적으로 업무핵심적인 부분에 대한 지원을 책임져야 한다.

클러스터링 기법은 이러한 업무 핵심적인 분야에서 응용 서버의 중단 시간을 줄이고 범위성(Scalability)을 향상시킴으로써 대용량의 클라이언트 요청에 대해서도 안정적으로 처리가 가능하도록 하는 서버 측 기술이다[2].

따라서, EJB 기반의 응용서버들이 업무핵심적인 분야에 있어서 중단 없는 서비스와 대용량 클라이언트 요청을 안정적으로 처리하기 위해서는 시스템 차원에서 클러스터링을 지원해야 한다.

본 논문에서는 EJB 서버에 대해 클러스터링 기술을 적용하기 위해 고려할 사항들과 이에 대한 구체적인 설계 사항에 대해 설명한다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문에선 구현한 클러스터링 토폴로지와 고려사항에 대해 설명하고, 3절에서는 각 구성요소들을 모듈별로 설계한 사항을 설명한다. 4절에서 결론을 맺는다.

2. 응용서버의 클러스터링 토폴로지

EJB 기반 응용서버의 클러스터링 토폴로지는 크게 세 가지로 구분할 수 있다. 첫 번째는 웹 서버와 응용서버가 하나의 노드를 구성하는 방식이다. 여기에서 웹 응용서버는 웹 컨테이너와 EJB컨테이너를 포함하는 개념이다. 이 방식에서 부하분배는 하드웨어에 의해 수행된다. 장점은 HTML, JSP/Servlet, EJB빈이 동일한 노드에 존재함으로 최적의 속도를 제고하고 관리가 용이하다는 점이다. 단점으로는 정적인 노드 관리만 지원함으로 확장성이 좋지 않다는 점이다. 두 번째는 웹 서버와 웹 응용서버가 분리된 모델이다. 이 모델에서 부하분배 및 고장복구 기능은 웹 서버에 의해 수행된다. 장점은 JSP/Servlet이 동일 노드에 존재함으로 EJB빈을 빈번이 호출하는 응용에서는 성

능이 뛰어나고 관리가 용이하다는 점이다. 실제 응용에서는 이러한 모델이 가장 많이 사용하고 있다. 세 번째는 웹 서버와 웹 컨테이너 및 EJB 컨테이너가 모두 분리된 모델이다. 부하분배와 고장복구 기능이 웹 서버와 웹 컨테이너에서 모두 수행된다. 장점은 확장성이 뛰어나다는 점이다. 단점으로는 여러 단의 노드에서 응용이 수행되므로 성능이 떨어지고 관리가 어렵다는 점이다[2].

따라서, 본 논문에서는 클러스터드 EJB 서버의 토폴로지는 두 번째 모델에 의해 구현하였으며, 구성 방식은 그림 1과 같다.

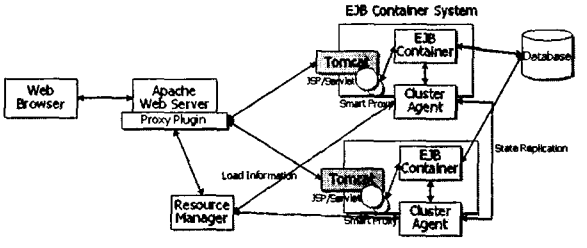

그림 1. 클러스터링 토폴로지

토폴로지 모델 두 번째를 사용해서 EJB 기반의 응용서버를 클러스터링으로 확장할 때 고려해야 할 사항은 그림 2와 같이 크게 6가지로 구분할 수 있다.

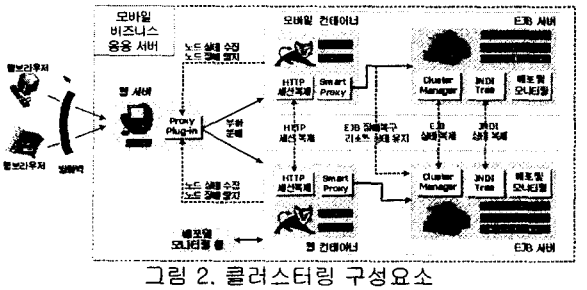

그림 2. 클러스터링 구성요소

첫 번째는 부하분배에 관한 문제이다. 웹 서버 단에서 각 웹 응용서버 측으로 부하를 분배하기 위해서는 소프트웨어 부하분배기가 필요하며, 웹 응용서버 단의 톰캣(Tomcat)과 같은 웹 컨테이너와 통신하거나 고장복구의 기능을 위해서 웹 서버에 플러그인 된 형태의 자바기반의 부하분배기가 존재하여야 한다. 두 번째는 웹 컨테이너 단에서 생성되는 HTTP 세션에 대해 각 노드에 복제하는 기능이 요구된다. 세 번째는 Servlet/JSP에서 EJB 빈을 호출할 때 사용되는 EJB빈의 Home과 Remote 스텁 단에서 EJB 빈의 메소드 호출에 대한 고장복구와 로컬 호출에 대한 최적화의 기능을 수행해야 된다. 이를 위해서는 기존의 스텁이 새로운 형태의 스텁이 생성되도록 해야 되며 본 논문에서는 이러한 스텁을 스마트 스텁(Smart Stub)이라 한다. 네 번째는 상태(Stateful) 세션 빈에 대한 상태 복제 기능이다. 다섯 번째는 각 노드에 존재하는 네이밍 서버의 정보 복제 기능이다. 여섯 번째는 구동 되고 있거나 또는 구동 준비 중이 노드에 개발자가 작성한 응용을 동일하게 서비스 중단 없이 배포하는 기능이다. 다음 절에서는 상기 여섯 가지 사항에 대해 설계된 사항을 설명한다.

3. 모듈별 설명

3.1 부하분배 모듈

그림 3은 상기에서 설명한 부하분배 고려사항에 대한 구현 블록도이다.

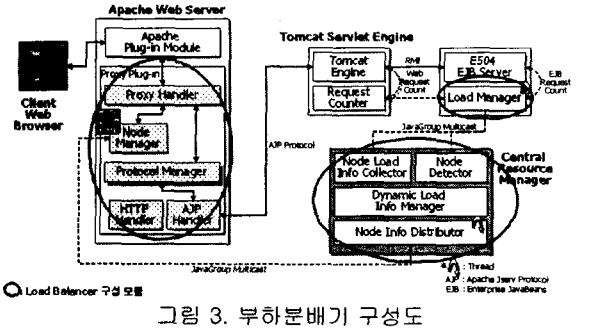

그림 3. 부하분배기 구성도

부하분배기는 자바로 구현되었으며, 아파치 웹 서버에 플러그인 되어 부하분배의 기능을 수행한다. 웹 서버 단에서 고장 복구의 기능을 위해서는 Idempotent URL에 대한 정의가 필요하며, 지정된 URL에 대해서는 부하분배기가 웹 컨테이너 단으로부터 전달된 예외 상황에 따라 적당한 다른 서버로 호출을 재 전송하게 된다. 동적인 부하 분배를 위해서 부하분배기는 각 노드의 부하정보를 실시간에 모니터링하고 이를 기반으로 부하 분배를 수행해야 된다. 이를 위해 중앙 노드 관리자는 부하분배기와 별도의 프로세스로 존재하고 각 노드의 부하정보를 실시간에 수집, 계산하여 부하분배를 위한 웨이트(Weight) 값을 생성한 후 부하분배기에 테이블 형태로 웨이트 값들을 제공한다.

3.2 HTTP 세션 복제 모듈

HTTP 세션은 JSP/Servlet에서 사용되는 세션 정보로써 개발자에 의해 웹 컴포넌트를 사용하는 사용자를 구분하기 위한 정보이다. 사용자 요청은 부하분배기에 의해 지속적으로 처리 노드가 변경됨으로 특정 노드에 생성되거나 변경되는 세션 정보는 각 노드로 복제되어야 한다. 복제를 위한 방법으로 데이터베이스를 사용하는 방법이 있으나 이는 실행 속도에 치명적인 영향을 줄 수 있으므로 본 논문에서는 인메모리 형태의 복제 방식을 사용한다. 그림 4는 HTTP 세션 복제 모듈의 블록도이다. 그림 4에서 JSP/Servlet은 기존의 HTTP 세션을 재구성한 Replicated Session을 사용하며, 이 세션은 웹 컴포넌트에 의해 생성되거나 상태가 변화될 때 마다 InMemory ReplicationManager에 의해 스트림 형태로 변환되어 각 노드에 전달된다. 또한 다른 노드로부터 전달된 스트림은 InMemory ReplicationManager에 의해 복원되어 해당 세

션에 상태가 반영된다.

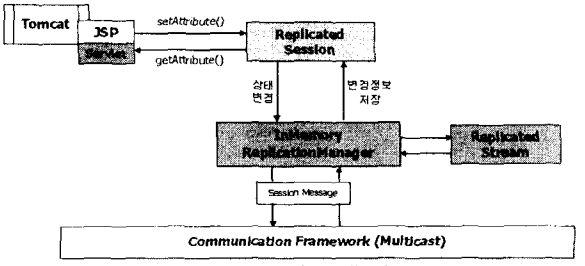

그림 4. HTTP 세션 상태 복제 모듈

3.3 스마트 스텁 모듈

스마트 스텁의 기능은 두 가지로 구분할 수 있다. 첫 번째는 특정 노드에서 시작된 트랜잭션에 대해 지역화하는 기능이다. 이 기능은 빈에서 사용하는 트랜잭션과 데이터베이스 연결 같은 리소스 정보를 노드마다 복제하지 않고, 클라이언트 요청이 다른 노드로 분배되었을 때 이전 노드에서 사용하던 리소스가 있을 경우 해당 노드로 요청을 재 전달하는 기능이다. EJB 컨테이너에서 사용하는 리소스 정보에 대한 복제는 상태 동기화를 통한 이득보다 복제에 대한 부하가 더 크기 때문에 동기화 대상에서 제외하였다. 두 번째 기능은 로컬 호출 최적화 기능이다. 스텁에서 빈을 호출하는 경우 기존의 일반적인 스텁에서는 소켓을 통한 리모트 호출이 이루어지나 본 논문의 스마트 스텁은 같은 VM(Virtual Machine)에 있는 빈을 호출할 경우 메모리에 존재하는 인스턴스를 직접 호출하므로 성능이 향상되는 장점이 있다.

3.4 EJB 빈 상태 복제 모듈

EJB 빈은 세션 빈, 엔터티 빈, 메시지 빈의 세가지 형태로 존재한다. 이중에서 상태 복제의 대상은 상태 세션 빈이다. 본 논문에서 엔터티 빈은 트랜잭션 지역화와 연계되어 호출이 있을 때 혹은 트랜잭션이 시작되거나 종료될 때 데이터베이스에 읽고 쓰는 작업을 수행함으로 동기화에 대한 필요가 없다. 무상태 세션 빈과 메시지 빈은 빈에 상태가 없으므로 동기화에 대한 필요가 없다. 따라서 본 논문에서는 상태 세션 빈에 대해서만 동기화를 고려한다. 클라이언트 요청이 상태 세션 빈을 호출하여 빈의 상태가 변경될 경우 인스턴스 관리 측의 상태 동기화 모듈이 이 정보를 각 노드에 전달하는 기능을 수행한다. 또한 전달된 동기화 정보는 해당 빈이 재 호출되는 시점에서 상태 정보를 동기화하여 클라이언트에 일관된 상태 정보를 제공한다.

3.5 네이밍 서버 상태 동기화 모듈

네이밍 서버는 각 노드에 존재하는 네이밍 서버들을 어떠한 방식으로 구성하느냐에 따라 동기화에 관한 방식이 결정된다. 즉, 네이밍 서버의 구성 방식은 크게 두 가지로 구분할 수 있다. 첫 번째는 중앙집중식 방식이다. 각 노드에 네이밍 서버를 별도로 구성하는 것이 아니라 클러스터에 대해 하나의 네이밍 서버 만을 구동하는 방식이다. 이 방식은 관리가 용이하고 동기화가 필요 없지만 네이밍 서버의 고장 시 모든 서비스가 중단되는 단점이 있다. 두 번째는 각 노드에 네이밍 서버를 별도로 구동하고 이들을 동기화하는 방식이다. 동기화를 해야 된다는 단점이 있고 유지 보수 비용이 많이 들지만 확장성과 가용성이 뛰어나다는 장점이 있다[3]. 본 논문에서는 두 번째 방식을 채택하여 단일장애점(Single Point Of Failure)를 최소화하였으며, 각 노드에 존재하는 네이밍 서버에 동기화를 위한 모듈을 추가하였다. 실제로 네이밍 서버에서의 동기화는 네이밍 서버에 객체를 바이딩하는 시점에서 발생하며, 이러한 경우는 응용이 배포되어 서비스를 준비하는 시점에만 발생함으로 실행시간의 동기화에 부하는 적다고 할 수 있다.

3.6 배포 모듈

클러스터를 구성하는 노드들이 서비스를 하고 있는 경우에 새로운 노드가 클러스터에 참여하려면 이전에 배포된 응용을 모두 전달 받아서 서비스 준비를 해야 한다. 또한 클러스터를 구성하는 노드들이 서비스를 준비하는 단계에서도 하나의 노드에 응용을 배포하면 이 응용은 클러스터의 모든 노드에 단일하게 배포 되야 한다. 각 노드에 존재하는 배포 모듈은 자신의 고유한 배포 저장소를 가지고 있으며, 개발자가 작성한 응용을 해당 노드의 저장소에 배포할 경우 이를 감지하여 클러스터에 참여하고 있는 모든 노드에 응용을 배포하도록 메시지를 전달한다. 메시지를 받은 노드들은 FTP를 통해 메시지를 전달한 노드로부터 응용을 다운받고 서비스 준비를 하게 된다. 또한 새로운 노드가 클러스터에 참여할 경우에는 이전에 클러스터에 참여하고 있던 노드들이 배포된 응용들의 정보를 메시지로 전달하고 새롭게 참여한 노드는 상기의 FTP 과정을 통해 응용을 서비스할 준비를 하게 된다. 여기에서 클러스터 노드간 주고 받는 메시지들은 모두 IP 멀티캐스트를 통해 수행되며, 응용서버에서 메시지를 주고받기 위한 통신 프레임웍을 제공한다.

4. 결론

본 논문에서는 EJB서버를 클러스터 환경으로 구성할 때 고려해야 토폴로지와 EJB 서버 측에서 구현해야 할 6가지의 모듈에 대해 살펴보았다. 본 논문의 토폴로지는 현재 상기에서 설명한 모델 중 두 번째만을 고려하여 구성하였으나, 보다 다양한 응용을 수용하기 위해서 토폴로지의 다변화가 이루어져야 한다. 또한 향후 EJB 서버의 성능 테스트 모델인 ECPerf(Enterprise Component Performance testing kit)에 의한 클러스터링 환경에서의 성능 테스트가 이루어져야 할 것이다.

[참고문헌]

[1] Linda G. Demichiel 외 2인, " Enterprise JavaBeans Specification, Version 2.0," Final Release, Sun Microsystems, 2001

[2] Abraham Kang, " J2EE Clustering" , http://www.Javaworld.com/jw-02-2001/jw-0223-extremescale.html, 2001

[3] Tyler Jewell. " EJB 2 Clustering with Application Servers" , http://www.oreillynet.com, 2000