모바일 응용 서버를 위한 클러스터링 환경 지원 구현

장철수 O, 김수형, 노명찬, 김중배

한국전자통신연구원 인터넷컴퓨팅연구부

{jangcs O, lifewsky, mcroh, jjkim }@etri.re.kr

An Implementation of the Clustering Environment for the Mobile Application Server

Choulsoo Jang O, Soohyung Kim, Myeongchan Roh, Joongbae Kim

Internet Computing Department,

Electronics and Telecommunications Research Institute

요 약

최근 이동통신 기술의 발전에 따라 개인용 컴퓨터뿐만 아니라 다양한 유형의 모바일 단말기까지 포함하는 유무선 통합 인터넷 서비스 환경으로의 변화가 급격히 일어나고 있다.유무선 통합의 인터넷 환경을 지원하는 모바일 응용 서버는 모바일 단말기의 제약 및 무선 통신과 관련된 제약 사항을 극복하면서, 휴대폰 및 PDA, PC 등 다양한 단말기를 타겟으로 한번 저작된 동일한 컨텐츠를 단말기 종류에 상관없이 서비스가 가능한 시스템으로 무선 환경에서의 효과적인 서비스를 제공할 수 있다. 이와 같은 모바일 응용 서버는 다수의 클라이언트로부터 전달된 대규모의 서비스 요청을 효율적으로 분산시켜 과부하를 방지하고 좀 더 원활한 서비스를 제공하기 위해 서비스 요청을 다수의 서버들에 분산시켜 모바일 응용 서버 시스템을 다중 서버 노드들로 구성 가능하도록 클러스터링 환경을 지원할 수 있어야 한다. 본 논문에서는 클러스터링 프레임웍을 사용한 모바일 응용 서버의 클러스터링 환경에 지원에 대한 설계 내용 및 구현 상황에 대해 논하겠다.

1. 서론

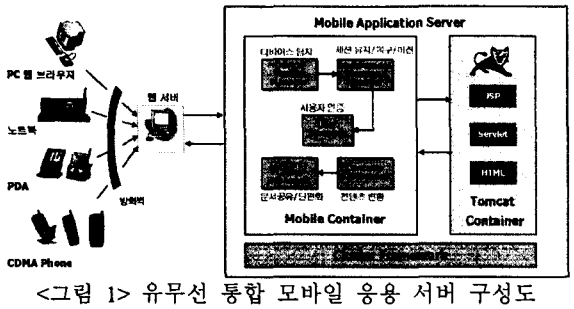
이동통신 기술의 발전에 따라 휴대폰 및 PDA 등 다양한 모바일 단말기를 이용한 무선 인터넷의 사용이 폭발적으로 증대되고 있으며, 이에 따라 기존 PC 를 타겟으로 한 유선 중심의 인터넷 서비스 환경이 다중 단말기를 대상으로 하는 유무선 통합의 인터넷 서비스 환경으로 급격하게 변하고 있다. 모바일 응용 서버는 모바일 단말기의 특징인 간헐적 단절성 및 컨텐츠 표현의 제약성 등을 극복하며 메타언어 기반의 단일 저작으로 다양한 단말기에 서비스가 가능한 모바일 환경 적응 미들웨어이다. 이와 같은 모바일 응용 서버는 다수의 클라이언트로부터 유무선을 통해 전달된 대규모의 서비스 요청을 효율적으로 분산시켜 고성능의 시스템 성능을 보장하기 위한 클러스터링 기술을 사용한다. 본 논문에서는 클러스터링 프레임웍을 사용한 모바일 응용 서버의 클러스터링 환경에 지원에 대한 설계 내용 및 구현 상황에 대해 살펴본다. 이 논문의 구성은 다음과 같다. 2 장에서는 유무선 통합의 모바일 응용 서버의 개념에 대해서 설명하고, 3 장에서는 클러스터링 환경지원에 사용되는 클러스터링 프레임웍에 대해 살펴보며, 4 장에서는 클러스터링 환경을 고려한 모바일 응용 서버의 설계 및 구현 내용에 대해 설명하고, 5 장에서 결론을 맺는다.

2. 유무선 통합 모바일 응용 서버의 개념

모바일 서비스에 대한 요구가 증가함에 따라 특정 단말기 형태만을 타겟으로 하여 모바일 서비스가 개발되고 있지만, 무선 인터넷 서비스를 효과적으로 제공하려면 단일의 저작으로 여러 형태의 단말기를 지원하여 각 단말 종류별로 재개발하지 않더라도 모바일 환경에 쉽게 적용되도록 하기 위한 유무선 통합 모바일 지원 기술 개발이 절대적으로 필요하다[1].

모바일 응용 서버는 기존 e-비즈니스 응용을 m-비즈니스 환경으로 쉽게 연동 확장할 수 있는 유무선 통합의 웹 컨테이너 시스템이다. 모바일 응용 서버는 클라이언트와 기존 Tomcat 서버와 같은 웹 컨테이너 사이의 중간 Tier 역할을 하여 선언적 방법으로 JSP, 서블릿 등의 웹 컴포넌트들을 모바일 환경에 적응시키는 기능을 제공하고 있다. 모바일 응용 서버의 구성은 <그림 1>과 같이 로깅이나 모니터링, 환경 설정 및 각종 컨테이너를 구동시키는 모바일 응용 서버 내에, JSP와 서블릿 같은 응용 페이지를 구동시키는 Tomcat 서블릿 컨테이너와 클라이언트의 요청을 가로채서 모바일 환경에 적응시키기 위해서 Tomcat 의 Valve[2] 형태로 구현된 모바일 컨테이너로 나눌 수 있으며 클러스터링 환경에서 모바일 컨테이너와 Tomcat 컨테이너의 상태를 복제하기 위한 클러스터링 프레임웍으로 구성

된다.

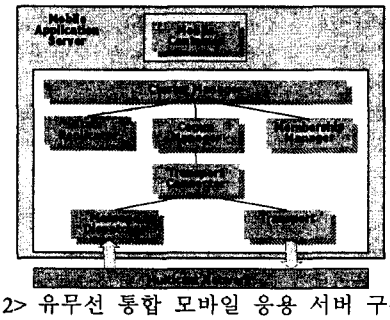


<그림 1> 유무선 통합 모바일 응용 서버 구성도

모바일 환경 적응을 위한 모바일 컨테이너에는 디바이스 프로파일 관리 모듈, 메타언어 기반의 단일 저작으로 다양한 단말기의 컨텐츠 표현의 제약성 등을 극복하기 위한 컨텐츠 변환 모듈, 컨텐츠의 공유 및 문서 단편화 기능을 제공하는 컨텐츠 캐쉬 모듈, 레거시 유저 관리자와 통합 API 를 제공하며 휴대폰과 같은 사용자가 인증된 단말기의 경우 자동 로그인 기능을 제공하는 사용자 관리 모듈, 모바일 환경의 간헐적 단절 투명성을 보장하며 다중 단말기간 세션 정보 자동 이전 기능을 제공하는 멀티 모달 세션 관리 모듈 등으로 구성되어 있다.

3. 클러스터링 프레임웍

모바일 응용 서버를 클러스터링 환경으로 구성하는 목적은 다수의 클라이언트로부터 전달된 대규모의 서비스 요청을 효율적으로 분산시켜 과부하를 방지하고 좀 더 원활한 서비스를 제공하기 위해 서비스 요청을 다수의 서버들에 분산시키기 위해서이며, 또한 하나의 서버가 다운되는 경우에도 서비스가 중단 없이 안정적으로 제공되기 위해 서비스를 클러스터 내 모든 서버에서 제공하기 위함이다[3]. 이와 같은 기능을 제공하기 위해 모바일 응용 서버에서의 클러스터링 프레임웍은 <그림 2> 와 같은 구성으로 이루어져 있다.



<그림 2> 유무선 통합 모바일 응용 서버 구성도

클러스터링 프레임웍의 클러스터 매니저는 클러스터에 필요한 기본 정보 및 관련 객체들을 관리 하며 모바일 응용 서버의 클러스터링 기능 제공 모듈들에 노드 정보와 노드 간 상태 복제와 관련된 API 를 제공한다. 클러스터 매니저는 Communication Manager, Membership Manager, State Replication Manager 를 두고 있다. Membership Manager 와 State Replication Manager 는 필요한 통신 기능을 얻기 위해 Communication Manger 로부터 통신 채널을 얻어오며, State Replication Manager 는 상태 복제 대상 노드를 결정하기 위해 Membership Manager 로부터 클러스터 구성 노드들에 대한 정보를 가져온다. Communication Manager 는 클러스터에 속한 전체 노드들에 대한 정보 전달을 위해 기본적으로 멀티캐스트 채널을 관리하지만 특정 노드에 대한 정보 전달을 위해 유니캐스트 채널을 관리할 수 있다. Transport 모듈과 연결되는 멀티캐스트 네트웍은 빠른 속도를 제공하기 위해서 Java SDK 의 IP Multicast 를 사용할 수도 있으나, 신뢰성이 다소 떨어지는 멀티캐스트 특성상 보다 높은 신뢰도를 요구하는 경우에는 JavaGroups [4]을 사용할 수 있도록 선택적으로 지정 가능하다.

4. 모바일 응용 서버의 클러스터링 환경 지원

위에서 설명한 바와 같이 모바일 응용 서버는 응용 로직의 실행을 위하여 Tomcat 서블릿 컨테이너를 포함하고 있으며, Tomcat 컨테이너의 밸브 형태로 모바일 환경 적응을 위한 모바일 컨테이너가 Tomcat 의 각 Context 마다 존재한다. 따라서 모바일 응용 서버는 클러스터링 프레임웍을 이용하여 Tomcat 의 세션 모듈과 모바일 컨테이너의 멀티모달 세션 모듈 및 컨텐츠 캐쉬 모듈에 대한 동기화 작업을 수행하도록 구현되어 있다. 4 장에서는 클러스터링 프레임웍을 사용한 클러스터링 환경 지원을 위한 구조에 대하여 설명한다.

Tomcat 은 다양한 레벨의 컨테이너들로 구성되어 있다. 즉, Engine, Host 및 Context 와 같이 3 개의 계층의 Container 로 이루어져 있으며, 최하위 컨테이너인 Context 마다 세션 관리자가 인스턴스화 되어 서비스된다. 클러스터링을 제공하기 위해서는 클러스터에 참여하는 Tomcat 노드간에 Session 의 복제가 이루어져야 하며, Tomcat 은 기본적으로 멀티캐스트를 이용하여 세션에 대한 클러스터링 환경을 제공한다. 이를 위해 Tomcat 에서는 DistributedManager 라는 세션 관리자를 제공하며, Tomcat 의 설정 파일인 server.xml 내의 <Context> Element 내의 <Manager> Element 에서 세션 관리자를 DistributedManager 로 대체하면 된다. 또한, Filip Hanik[5]는 멀티캐스트 프레임웍인 JavaGroups 을 이용하여 Tomcat 에서의 In Memory 상태의 세션 복제를 할 수 있는 InMemoryReplicationManger 을 구현하였다. DistributedManager 및 InMemoryReplicationManger 는 모두 Context 내에서만 동작한다. 그러나, 모바일 응용 서버는는 Context 의 범위를 넘어서 Engine 레벨의 클러스터링을 고려한다. Engine 레벨의 클러스터링을 고려하는 이유는 각 Context 레벨 단위마다 클러스터링 프레

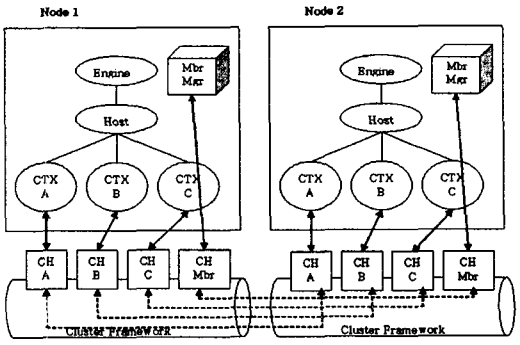
임웍의 인스턴스를 관리할 때에는 다음과 같은 단점이 있기 때문이다.

- 같은 Server 의 Context 끼리는 서로 정보를 복제할 필요가 없다.
- 각 Context 별로 클러스터링 프레임웍을 인스턴스화시키면 많은 메모리 및 CPU 를 점유하여 전체적인 성능을 저하시킨다.
- Context 레벨에 각각의 멤버쉽 관리자가 존재하면 중복되는 많은 메시지가 네트웍에 유포되어 네트웍의 효율을 떨어뜨리며, 또한 이들 메시지를 처리하기 위해 많은 불필요한 자원(CPU, 메모리)의 낭비를 가져올 것이다.

이와 같은 이유로 Context 레벨이 아닌 Engine 레벨에서 멀티캐스트 프레임웍을 제공하여 클러스터링 환경을 지원하여야 한다.

또한, Engine 레벨에서 멀티캐스트 프레임웍을 통해 전달 받은 세션 변경 메시지를 어떤 방법으로 하위 레벨의 Context 에 전달하며, 어떻게 해당 Session Manager 가 세션 변경 메시지에 따라 자신이 관리하고 있는 세션 정보를 변경하도록 할 것인가를 고려하여야 한다. 한가지 방법은, 상위레벨(Engine)에서 하위레벨(Context)까지는 findChild(), findChildren() 등의 메쏘드를 이용하여 하위레벨까지 찾아가서 특정 기능을 호출하는 방법이 있겠다. 이 방법에서는 정보를 복제하기 위해서 특별한 메쏘드가 하위레벨(Context)에 추가되어야 하며 이렇게 추가된 메쏘드들은 정보 복제를 위해 호출되어야 한다. 그러나, 이렇게 되면 Tomcat Container 의 소스를 수정해야 하므로 피해야 할 방법이다. 따라서, 상위레벨에서 하위레벨을 호출하는 첫번째 방법과는 반대로 하위레벨(Context)에서 상위레벨(Engine)의 멀티캐스트 프레임웍을 접근하여 복제 메시지를 주고 받을 수 있도록 하는 방법이 적당하다. 그러나, 이 방법에서는 각 Context 마다 직접 멀티캐스트 프레임웍을 접근하여 메시지를 읽으려고 하면, 공유된 멀티캐스트 프레임웍에 대한 locking 과정이 있어야 하므로 프레임웍에 대한 lock 을 얻지 못한 다른 Context 는 대기상태로 블록되어 병목현상이 발생할 수 있다. 이러한 병목현상을 제거하기 위해서 Context 에서는 멀티캐스트 프레임웍을 직접 접근하지 않고, <그림 3>과 같이 멀티캐스트 프레임웍에 각 Context 별로 Channel 을 두어 해당 Channel 만을 접근하도록 하였다.

이러한 구조를 구현하기 위해서는 모바일 응용 서버가 메인이 되어 Tomcat 컨테이너와 클러스터링 프레임웍을 인스턴스화시켜야 하며, Tomcat 의 Context 레벨의 밸브 형태로 존재하는 모바일 컨테이너가 클러스터링 프레임웍으로부터 해당 Context 의 Channel 을 얻고 각 모듈의 복제 메시지를 주고 받도록 하고 있다. Tomcat 의 세션 복제와 마찬가지로 모바일 컨테이너의 캐쉬 관리 모듈과 멀티 모달 세션 관리 모듈은 클러스터링 프레임웍을 통해서 노드간 복제를 통해 상태정보를 동기화하고 있다. 그림상에는 Membership Manager 는 모바일 응용 서버 내부에 별도로 존재하는 것처럼 그려져 있으나, Membership Manager 는 3 장에서 설명한 바와 같이 모바일 응용 서버에서 인스턴스화 시킨 클러스터링 프레임웍 내에 이미 구현되어 존재하고 있다. 따라서, Membership Manager 에 대한 별도의 구현은 하지 않아도 된다



<그림 3> 클러스터링 지원을 위한 모바일 응용 서버

5. 결론

클러스터링 환경으로 확장된 모바일 응용서버에서 클러스터링 프레임웍을 이용한 구조에 대해 살펴보았다. 그러나, 멀티캐스트를 이용한 클러스터링 방법은 하나의 클러스터에 서버 노드들을 2~3 개 정도로 구성하는 경우에는 적당하지만 수십대의 클러스터링 환경을 구성한다면, 노드간에 주고 받는 메시지가 폭주하여 클러스터링 프레임웍 자체가 병목구간이 되어 전체 성능을 저하시키는 요인이 될 수 있다. 따라서, 중소규모의 mission critical 한 업무에 대해, 일정한 정도의 성능을 향상시키기에는 적당하겠지만 대규모의 클라이언트 요청을 처리하기에는 부적당할 수 있다. 따라서, 향후 연구에서는 멀티캐스트와 중앙 DB 를 혼용한 방식의 클러스터링 프레임웍을 통해 메시지 교환을 최소화하여 클러스터링 환경을 제공할 수 있는 해결책을 찾고자 한다.

참고문헌

[1] 김성훈외 7 인, 유무선 통합 모바일 응용 서버에 관한 연구, 정보과학회학술지, 제 20 권 1 호, 2002

[2] Apache group, "Server Configuration Reference – The Valve Component", http://jakarta.apache.org/tomcat/tomcat-4.0-doc/config/valve.html

[3] 김수형외 3 인, "Clustered EJB 서버에서의 멀티캐스트 프레임워크 연구", 한국정보처리학회 추계학술대회 논문집 제 9 권 제 2 호, 2002

[4] Bela Ban, "JavaGroups User's Guide," http://www.javagroups.com/javagroupsnew/docs/newuser.zip

[5] Filip Hanik, "In Memory Session Replication In Tomcat 4",http://www.theserverside.com/resources/article.jsp?l=Tomcat