Oracle9*i* Application Server

자바2 Enterprise Edition 기능 및 디자인 고려사항

2001년 5월

ORACLE®

Oracle9*i*AS v1.0.2.2는 오라클의 자바 Strategy 구현을 위한 커다란 진전을 의미합니다. 이 릴리스는 모두 자바로 작동되어 표준 자바 Development Kit(JDK) Virtual Machine(자바 VM)에서 실행되는 완벽한 자바2 Enterprise Edition(J2EE) 컨테이너를 제공합니다. Oracle9*i*AS Containers for J2EE(OC4J)가 제공하는 주요 기능은 다음과 같습니다.

JDK에서 실행되는 순수 자바 컨테이너/런타임

먼저, OC4J는 완전히 자바로 구현됨에 따라 다음과 같은 장점을 발휘합니다.

1) 경량화 15Mb 디스크, 20Mb 메모리
2) 신속한 설치 5분 이내
3) 간편한 사용 표준 자바 Development 및 Profiling 툴 지원
4) Solaris, HP-UX, AIX, Tru64, Windows NT 및 Linux와 같은 모든 표준 운영 체제 및 하드웨어 플랫폼에서 사용가능

Oracle9*i*AS v1.0.2.2는 JDK 1.2.2와 JDK 1.3 자바 VM에서 실행되도록 검증되었습니다.

완벽한 J2EE 1.2 컨테이너

둘째, OC4J는 JSP Translator, 자바 서블릿 엔진과 Enterprise JavaBeans(EJB) 컨테이너와 같은 완벽한 J2EE 컨테이너를 제공합니다. 또한 자바 Messaging Service, JMS 및 기타 몇 가지 자바 사양 또한 지원합니다. Oracle9*i*AS v1.0.2.2는 J2EE가 인증했으며 다음과 같이 모든 J2EE API를 완벽하게 지원합니다.

· Enterprise 자바 Beans(EJB) 1.1 및 EJB 2.0형 Enterprise Beans와 Message Driven Beans와 같은 몇 가지 EJB 2.0 사양
· Servlet 2.2(Servlet Chaining and Filters 기능과 같은 Servlet 2.3 포함)
· 자바 Server Pages (JSP) 1.1
· 자바 Transaction API (JTA) 1.0.1
· 자바 Naming and Directory Interface (JNDI) 1.2
· 자바 Messaging Service (JMS) 1.0.1
· 자바 Database Connectivity (JDBC) 2.0
· 자바Mail 1.1.2

또한 OC4J는 J2EE 1.3 사양의 일부인 J2EE 1.3 사양의 구현을 제공합니다. 또한 JSP Tag 라이브러리, WAR 및 EAR 파일 기반의 배치와 같은 표준 서비스와 J2EE Applications의 자동 배치와 신속한 배치를 지원합니다.

경쟁력 있는 성능/확장성/가용성

Oracle9*i*AS v1.0.2.2는 개발자에게 다음과 같은 여러 가지 장점을 제공합니다.

1) 빠른 성능: OC4J는 업계에서 가장 빠른 자바 애플리케이션 서버입니다. 별도의 문서에서 업계의 주요 경쟁사와의 벤치마크 결과를 제공할 것입니다.
2) 확장성과 가용성: 훨씬 적은 메모리 용량으로도 가능합니다. 단일 장애 지점이 없으며 무상태(stateless)

및 상태유지(stateful) 애플리케이션 장애복구를 모두 처리하기 위해 정교한 IP-Multicast 기반 클러스터링 기능을 제공합니다.

또한 OC4J는 통합된 자바 기반 Management 콘솔을 제공하여 한 곳에서 컨테이너의 단일 인스턴스 또는 인스턴스 클러스터/그룹을 구성하고 감시하며 관리할 수 있는 기능을 제공합니다.

이 문서는 Oracle9*i*AS v1.0.2.2의 J2EE 기능에 대한 기술적인 개요를 제공합니다. 먼저 이 문서에는 오라클의 자바 Strategy와 Product Roadmap이 설명되며, 그 다음 J2EE의 개요가 설명되며 J2EE의 지원을 위한 Oracle9*i*AS v1.0.2.2의 특징이 설명됩니다. 마지막으로 Oracle9*i*AS Containers for J2EE의 확장성과 가용성 기능이 설명됩니다.

1. 서문

지난 2년 동안 자바는 웹 기반 애플리케이션 개발과 배치에 선택되어 사용된 언어입니다. 단순하고 낮은 원가의 운영 체제와 하드웨어 플랫폼과는 별도의 서버 기반 정보 기술 하부구조에서 실행할 수 있는 통일된 휴대형 애플리케이션 개발 솔루션에 대한 자바의 약속은 애플리케이션 개발자들과 기업의 정보관리 책임자(CIO)들로부터 상당한 관심을 불러 일으켰습니다. 이들은 고객, 직원 및 공급업체에 사용하기 쉬운 서비스를 제공함으로써 서비스의 범위를 넓히고 원가를 절감하며 응답 시간을 단축해야 합니다. 보통 이러한 서비스를 제공하는 애플리케이션은 기존의 정보 시스템을 광범위한 사용자에게 서비스를 제공하는 새로운 비즈니스 기능과 결합해야 합니다. 이들 서비스는 1) 가용성이 높아 최근 전세계 비즈니스 환경의 요구를 충족하고 2) 사용자의 개인 생활과 엔터프라이즈의 무결성을 보호하는 보안성을 갖추어야 하고 3) 비즈니스 트랜잭션을 정밀하고 신속하게 처리할 수 있는 신뢰성과 확장성을 갖추어야 합니다.

인터넷 애플리케이션은 멀티 티어 애플리케이션으로 디자인되고 구현됩니다. 미들 티어 애플리케이션 서버는 애플리케이션에 런타임 환경을 제공하며 기존의 비즈니스 기능, 레거시 시스템 및 데이타에 대한 액세스를 제공합니다. 자바 2 Platform, Enterprise Edition(J2EE)은 멀티 티어인 인터넷 애플리케이션 개발을 위해 표준 애플리케이션 모델과 프로그래밍 인터페이스 표준을 정의함으로써 이들 멀티 티어 애플리케이션 개발에 따른 원가와 복잡성을 줄여줍니다. J2EE를 지원하는 애플리케이션 서버는 J2EE 애플리케이션의 실행을 위한 표준 컨테이너 환경을 제공하므로 애플리케이션 개발자가 애플리케이션을 한번만 작성하면 여러 업체에게 서버 환경을 제공할 수 있습니다.

Oracle9*i*AS v1.0.2.2는 표준 자바 개발 키트(JDK) 가상 머신(자바 VM)에서 실행되는, 완전히 자바로 작성되고 빠르고 경량이며 확장성이 뛰어나고 사용하기 쉬운 자바2 Enterprise Edition(J2EE) 컨테이너를 제공합니다. Oracle9*i*AS v1.0.2.2는 전체 J2EE 1.2 Container에 Enterprise 자바 Beans(EJB) 1.1, Servlet 2.2, 자바 Server Pages(JSP) 1.1, JTA 1.0.1, JNDI 1.2, JMS 1.0, JDBC 2.0과 자바Mail 1.1.2를 위한 완벽한 지원을 제공합니다. 또한 J2EE Container는 J2EE 1.3 Specification의 일부인 EJB 2.0과 Servlet 2.3을 거의 완벽하게 구현합니다. 또한 JSP Tab 라이브러리 WAR 및 EAR 파일 기반 배치, J2EE Application의 배치와 즉시 배치와 같은 표준 서비스를 지원하며 무상태(stateless) 및 상태유지(stateful) Application Failover를 모두 지원하는 고급 클러스터링 기능을 제공합니다. 이 문서는 Oracle9*i*AS v1.0.2.2의 J2EE Facility에 관한 자세한 기술적인 개요를 제공합니다. 이 문서는 다음과 같

이 세 부분으로 구성됩니다.

· 첫째, 오라클의 자바 Strategy와 Product Roadmap

· 둘째, J2EE의 개요 및 Oracle9*i*AS v1.0.2.2의 지원 특성 설명

· 셋째, Oracle9*i*AS Container for J2EE 확장성 및 가용성 기능 개요

2. 오라클의 자바 전략 및 제품 로드맵

오라클은 자바에 많은 전략적 노력을 기울었으며 전사적으로 수많은 제품과 서비스에 자바를 통합 중입니다. 자바에 대한 오라클의 세 가지 목표는 다음과 같습니다.

· 오라클은 엔터프라이즈 자바 애플리케이션을 개발하고 배치할 목적으로 특히 자바2 Enterprise Edition(J2EE)을 표준 방법론으로 지원하는 데 모든 노력을 기울이고 있습니다.

· 오라클은 쉽고 생산적으로 애플리케이션 개발을 할 수 있도록 풍부한 자바 툴과 프로그래밍 인터페이스로 구성되는 완벽한 자바용 개발 환경을 자바 개발자에게 제공합니다.

· 오라클은 자바/J2EE 애플리케이션 배치를 위해 확장성과 가용성이 뛰어난 고성능의 서버 하부구조를 제공합니다.

아래 그림은 오라클의 자바 전략을 보여주고 있는데, 이 내용은 아래에 보다 상세히 설명되어 있습니다.

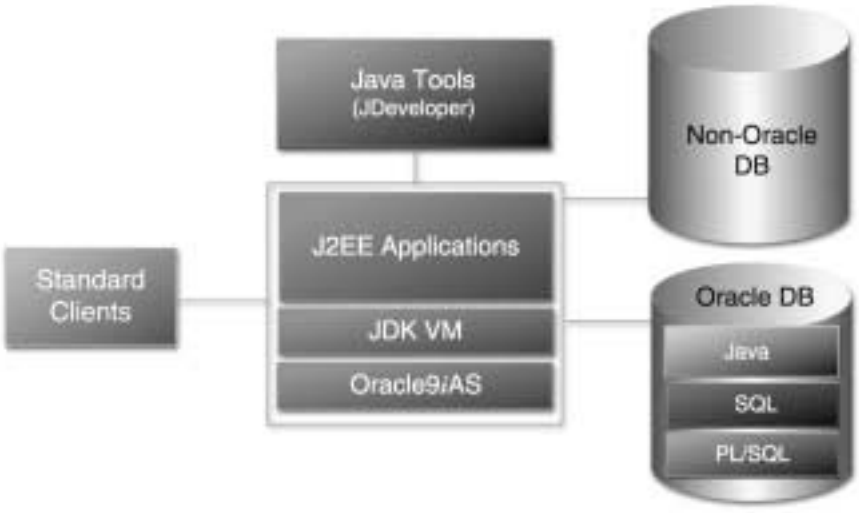


그림: 오라클의 자바 전략

· J2EE 지원: Oracle9*i*AS의 최신 릴리스는 인증된 J2EE입니다. Oracle9*i*AS v1.0.2.2는 모든 J2EE 1.2 표준을 지원하며 특히 서블릿 2.3과 EJB 2.0과 같은 새로운 J2EE 1.3 표준의 상당 부분을 이미 지원하고 있습니다(J2EE 1.3 표준은 아직 완성되지 않은 상태입니다. 오라클은 표준 위원회에 적극적으로 참여하고 있으며 이 표준이 릴리스 되는 즉시 지원할 수 있도록 만반의 준비를 하고 있습니다)

· 툴과 데이타베이스 액세스 인터페이스. 오라클은 JDeveloper를 자바 개발 툴(자바 IDE)과 JDBC 및 자바

코드 내의 내장 SQL(SQLJ)의 두 가지 데이타베이스 액세스 인터페이스로서 제공합니다.

- JDeveloper: Oracle JDeveloper는 J2EE Application의 개발과 배치를 100% 지원하는 생산적 개발 툴입니다. 또한 JDeveloper는 자바 클래스와 데이타베이스 스키마 사이의 일대일 및 일대 다수 매핑을 지원하며 질의성을 위해 SQL의 사용을 허용하는 자바의 비즈니스 컴포넌트인 오브젝트 연관 매핑을 통합합니다.
- JDBC 드라이버: 클라이언트-서버 자바 애플리케이션이나 미들 티어 자바 Application을 개발하는 개발자를 위한 Type 2 JDBC/OCI와 자바 애플릿을 작성하는 개발자를 위한 Type 4 Thin JDBC 드라이버, 이 두 가지의 JDBC 드라이버가 제공됩니다. JDBC의 특수 버전이 오라클 데이타베이스 안에서 실행되어 서버의 자바 VM에서 실행되는 자바 애플리케이션이 로컬로 정의된 데이타에 액세스할 수 있습니다. 또한 오라클은 Merant의 JDBC 드라이버를 사용하여 IBM/DB/2, Microsoft SQL-Server, Informix 및 Sybase와 같은 비 오라클 데이타베이스에 액세스할 것을 권장합니다(Merant는 오라클 파트너입니다).
- 자바 코드 내의 내장 SQL: 오라클은 또한 SQL문을 자바 프로그램에 내장하는 표준 메소드를 제공합니다. 이 메소드는 JDBC보다 훨씬 단순하고 생산적인 프로그래밍 API를 제공합니다. 사용자는 이 높은 수준의 API에 애플리케이션을 작성한 다음 표준 프로세스를 사용하여 JDBC 호출로 프로그램을 표준 자바 원본으로 변환합니다. 프로그램은 표준 JDBC 드라이버를 사용하여 복수 공급업체 데이타베이스와 통신할 수 있습니다. 자바 코드 내의 내장 SQL은 자바에서 데이타베이스에 액세스하는 클라이언트와 미들 티어 애플리케이션을 모두 개발하고 휴대형 방식으로 자바의 데이타베이스 서버 안에 저장된 절차, 트리거, 메소드 등을 아주 쉽게 정의하는 메소드를 제공합니다.

· 자바 서버 환경: 오라클은 사용자가 인터넷 애플리케이션 코드를 세 가지 계층, 즉 웹 프리젠테이션 로직, 비즈니스 로직 및 데이타 조작 로직으로 분할한다는 사실을 잘 알고 있습니다.

- J2EE 컨테이너: J2EE 애플리케이션에서 웹 프리젠테이션 로직은 자바 Server Page(JSP)나 서블릿을 사용하여 개발됩니다. 비즈니스 로직은 Enterprise 자바Bean(EJB)으로서 개발됩니다. Oracle9*i*AS는 J2EE Application을 지원하기 위해 JDK 자바 Virtual Machine(각 운영 체제와 하드웨어 플랫폼에 제공되는 자바 VM)에서 실행되는 표준 준수 J2EE 컨테이너를 제공합니다.

· 데이타베이스 자바 환경: 자바에서의 데이타 처리를 위해 오라클은 자바 가상 머신(JDK와 호환)과 서버 환경을 SQL 및 PL/SQL과 밀접하게 통합된 데이타베이스에 제공합니다. 사용자는 Oracle Database 환경을 사용하여 재래식 데이타베이스 저장 프로시저, 트리거 및 Abstract Data Type(ADT) 메소드를 자바로 작성할 수 있습니다. 사용자는 데이타베이스의 자바 Environment를 사용하여 자바를 프로그래밍 언어로 사용함으로써, 데이타를 미들 티어로 인출하기보다는 오라클 데이타베이스에서 데이타를 조작하여 애플리케이션 성능을 향상시킬 수 있습니다.

지금까지는 오라클의 자바 전략을 살펴보았으며, 이제는 Oracle9*i*AS Containers for J2EE를 살펴보고 Oracle9*i*AS v1.0.2.2의 J2EE 기능을 자세히 검토하도록 하겠습니다.

3. Oracle9*i*AS Containers for J2EE(OC4J)

Oracle9*i*AS v1.0.2.2는 완벽한 자바2 Enterprise Edition(J2EE) 환경에 Oracle9*i*AS Containers for J2EE(OC4J)를 제공합니다. OC4J는 완전히 자바로 작성되며 표준 Java Development Kit(JDK) Virtual Machine(자바 VM)에서 실행됩니다. OC4J의 주요 기능은 다음과 같습니다.

- JDK에서 실행하는 순수 자바 컨테이너/런타임: 먼저 OC4J는 완전히 자바로 구현됨에 따라 다음과 같은 장점을 발휘합니다. 1) 경량화 15Mb 디스크, 20Mb 메모리, 2) 신속한 설치 5분 이내, 3) 간편한 사용 표준 자바 Development 및 Profiling 툴 지원, 4) Solaris, HP-UX, AIX, Tru64, Windows NT 및 Linux와 같은 모든 표준 운영 체제 및 하드웨어 플랫폼에서 사용가능. Oracle9*i*AS v1.0.2.2는 JDK 1.2.2와 JDK 1.3 자바 VM에서 실행되도록 검증되었습니다.
- 완벽한 J2EE 1.2 컨테이너: 둘째, JSP Translator, 자바 서블릿 엔진 및 Enterprise JavaBeans(EJB) 컨테이너와 같은 완전한 J2EE 컨테이너을 제공합니다. Oracle9*i*AS v1.0.2.2는 Enterprise JavaBeans(EJB) 1.1, Servlet 2.2, 자바 Server Pages(JSP) 1.1, JTA 1.0.1, JNDI 1.2, JMS 1.0, JDBC 2.0 및 자바Mail 1.1.2와 같은 모든 J2EE 1.2 API에 대해 완벽한 지원을 제공합니다.

이 절에서는 Oracle9*i*AS Containers for J2EE에 대해 자세히 검토합니다. J2EE에 익숙하지 않은 독자를 위해 J2EE의 각 프로그래밍 인터페이스나 서비스를 각각 간략히 설명한 다음 그러한 서비스에 대한 OC4J의 지원 기능을 설명합니다. 아래의 그림은 Oracle9*i*AS의 J2EE의 기능을 요약한 내용입니다.

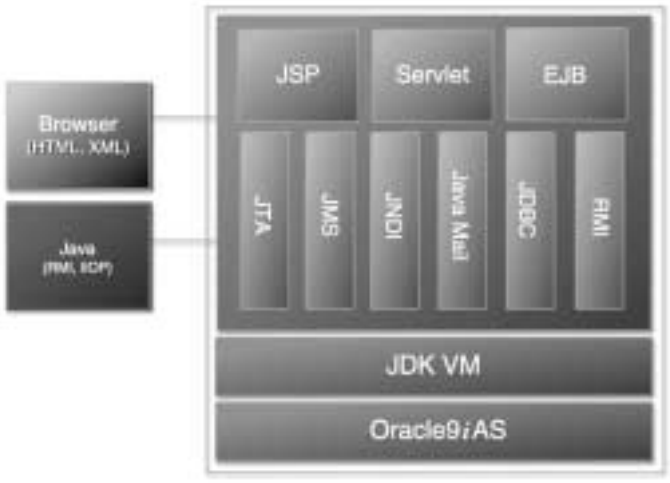


그림: Oracle9*i*AS Containers for J2EE v.1.0.2.2의 J2EE 기능

A. 자바 서블릿

자바 서블릿은 웹 서버 기능을 확대한 프로그램입니다. 서블릿은 클라이언트로부터 요청을 받아 동적으로 응답을 생성한 다음(요청을 만족시키기 위해 데이타베이스 질의) HTML이나 XML 문서가 수록된 응답을 클라이언트로 보냅니다. 서블릿은 CGI와는 비슷하지만 자바 클래스와 스트림을 사용하기 때문에 작성이 훨씬 쉽습니다. 또한 서블릿은 자바 Byte 코드와 호환되며 서블릿 인스턴스가 런타임으로 메모리에 저장되

어 각 클라이언트가 스레드를 만들기 때문에 실행 속도도 빠릅니다. 서블릿이 당면한 현안은 HTTP가 무상태(stateless) 프로토콜이라는 점입니다. 즉, 각 요청이 새로운 연결로서 수행되기 때문에 요청 사이에서 흐름 제어가 자연스럽게 이루어지지 않습니다. 세션 트래킹이나 세션 관리로 요청 간 특정 클라이언트 상태를 유지할 수 있습니다.

OC4J의 Servlet Engine은 JDK 자바 VM에서 실행되며 다음과 같은 내용을 제공합니다.

- Servlet 2.3 지원: OC4J Servlet Engine은 Servlet 2.2에 완벽한 지원을 제공하며 Servlet 2.3 사양(Servlet 2.3 사양은 앞으로 발표될 J2EE 1.3 사양의 일부)을 지원합니다.
- Tomcat와 100% 호환: OC4J Engine은 Apache 콘소시엄이 공급하는 Tomcat Servlet Engine과 100% 호환됩니다. 그 결과 애플리케이션 개발을 위해 Apache와 Tomcat를 사용한 개발자는 애플리케이션을 쉽게 Oracle9*i*AS에서 배치할 수 있습니다.
- 필터에 대한 완벽한 지원: Oracle9*i*AS v1.0.2.2는 Servlet 2.3 사양의 일부인 단순한 필터와 복잡한 필터를 지원합니다. 특히 필터는 URL 패턴이나 Servlet Name과 같은 것에 필터가 매핑되는 자원을 클라이언트가 요청할 때 호출되는 구성 요소입니다. 필터는 일반적으로 원래 요청 대상의 실행 전이나 후 요청, 응답 또는 헤더 값의 래핑 및 조작에 사용되며 그 자체로는 클라이언트에 응답하지 않습니다. 필터는 단일 요청에 대해 작용할 때에는 단순하게, 복수 요청에 대해 작용할 때에는 복잡하게 구성되며 입력과 출력 인수를 수신할 수 있는 곳에 대해 파라미터화 되고 차례로 여러 필터를 호출할 수 있는 체인에 연결됩니다.
- 완전한 WAR 파일 기반 배치: 서블릿은 Web Application Archive(WAR) 파일이라는 표준 형식을 사용하여 패키지되어 J2EE에 배치됩니다. OC4J는 1) 몇 개의 서블릿(JSP 포함. 아래 내용 참조)과 패키지를 WAR 파일에 포함시키는 WAR 파일 Packaging Tool, 2) WAR 결과를 선택하여 하나 이상의 Oracle9*i*AS에 배치하는 WAR 파일 Deployment Tool을 제공합니다. WAR Deployment Tool은 클러스터 배치로 지원하여 특정 아카이브를 클러스터 구성을 위해 정의된 모든 Oracle9*i*AS 인스턴스에 동시에 배치할 수 있습니다.
- 서블릿의 자동 컴파일, 자동 배치: 또한 OC4J는 서버가 WAR 아카이브를 받아 자동으로 압축 풀기를 하여 애플리케이션을 설치하는 서블릿의 자동 컴파일과 자동 배치 기능을 제공합니다. 이들 기능은 모두 J2EE 애플리케이션 구축의 개발, 컴파일 및 배치 주기를 실질적으로 단축합니다.
- 서블릿의 상태유지 장애복구(stateful failover) 및 클러스터 개발: 클러스터는 투명한 메소드로 확장성이 뛰어나고 가용성이 높은 서비스를 제공하기 위해 운영을 조정하는 Oracle9*i*AS 서버의 그룹입니다. 서블릿은 HTTP 세션 오브젝트를 사용하여 예를 들어 웹 장바구니나 여행 일정 컨텐트와 같은 메소드 요청 사이에 상태를 저장합니다. OC4J는 서블릿이 특히 HTTP 세션 오브젝트에 관한 서블릿 세션 상태를 투명하게(즉, 프로그램 API를 변경하지 않고) 다른 Oracle9*i*AS 인스턴스에 복제할 수 있는 IP 멀티 캐스트 클러스터링 메커니즘을 지원합니다(자세한 내용은 아래의 제3항 참조).

B. 자바 Server Page

자바 Server Page(JSP)는 텍스트 기반 프리젠테이션 중심의 서블릿 개발 방법입니다. 웹 개발자와 디자이

너는 JSP를 사용하여 기존의 비즈니스 시스템을 강화하는 풍부한 정보의 동적 웹 페이지를 신속하게 개발하여 쉽게 유지할 수 있습니다. JSP는 컨텐트 프리젠테이션을 컨텐트 생성에서 분리하여, 웹 디자이너가 기본적인 동적 컨텐트를 변경할 필요 없이 전체적인 페이지 레이아웃을 변경할 수 있도록 해줍니다. JSP는 자바 프로그래밍 언어로 작성된 XML과 같은 태그와 스크립틀릿을 사용하여 페이지의 컨텐트를 생성하는 로직을 캡슐화합니다. 또한 애플리케이션 로직은 페이지가 이러한 태그와 스크립틀릿으로 액세스하는 JavaBeans와 같은 서버 기반 자원에 상주할 수 있습니다. 모든 포매팅(HTML이나 XML) 태그는 직접 응답 태그 뒤로 전달됩니다. JSP 기술은 페이지 로직을 디자인과 디스플레이에서 분리하고 재사용할 수 있는 구성 요소 기반 디자인을 지원함으로써 웹 기반 애플리케이션을 훨씬 빠르고 쉽게 구축할 수 있습니다. JSP 페이지는 표준 HTML이나 XML 페이지와 비슷하며 여기에 JSP 엔진이 처리하고 스트립한 추가 요소가 추가됩니다. 일반적으로 JSP 요소는 결과 페이지에 삽입되는 텍스트를 만듭니다.

JSP 페이지와 서블릿은 Common Gateway Interface(CGI)보다 바람직합니다. 그 이유는 CGI의 경우 플랫폼 독립적이지 않고 오버헤드가 더 필요하며 파라미터 데이타를 액세스하여 프로그램에 전달할 수 있는 쉬운 방법이 없기 때문입니다. 애플리케이션 개발자가 JSP를 사용할 수 있는 방법이 몇 가지 있습니다.

1) JSP 페이지는 JavaBeans 클래스와 함께 사용하여 비슷한 모양과 느낌의 페이지로 구성된 웹 사이트 구축을 위한 웹 템플리트를 정의할 수 있습니다. JavaBeans 클래스를 사용하는 템플리트는 자바 코드가 없어도 데이타를 렌더링할 수 있습니다. 즉, HTML 편집기로 유지할 수 있습니다.

2) JSP 페이지를 사용하는 단순한 웹 기반 애플리케이션의 경우 컨텐트는 JavaBeans 클래스 대신 사용자 정의 태그나 스크립틀릿을 사용하여 애플리케이션 로직에 구속될 수 있습니다.

JSP 페이지는 다음과 같은 요소로 구성됩니다.

· JSP Directives: JSP Directives는 정보를 JSP 엔진에 전달합니다. JSP Directives에는 버퍼와 스레드 정보 또는 오류 취급과 같은 페이지 고유 정보를 전달하는 Page Directives, 확장자와 함께 스크립팅 언어를 지정하는 Language Directives, 페이지에 외부 문서를 포함시킬 때 사용되는 Include Directives 및 페이지가 호출할 수 있는 사용자 정의 태그의 라이브러리를 표시하는 Taglib Directive와 같은 몇 가지 유형이 있습니다.

· JSP 작업 또는 태그: JSP 고유 XML 기반 태그를 통해 JSP 처리를 구현합니다. JSP 페이지에서 사용할 수 있는 기능성의 캡슐화에 태그가 사용됩니다. 그 예로는 조건부 로직, 데이타베이스 액세스 및 반복을 들 수 있습니다. JSP 1.0에는 코어 태그라고 하는 많은 표준 태그가 있으며 JSP 1.1에는 표준 태그가 추가로 수록됩니다. 태그의 장점은 애플리케이션 간에 사용하기 쉽고 공유하기 쉽다는 데에 있습니다.

· 스크립틀릿: 또한 JSP 페이지에는 스크립틀릿이라고 하는 소형 스크립트가 포함됩니다. 스크립틀릿은 요청 시간 처리로 실행되는 코드 부분입니다. 스크립틀릿은 페이지(위의 예와 같이)의 정적 요소와 결합하여 동적으로 생성되는 페이지를 만들 수 있습니다. 스크립트는 〈% and %〉 마커 안에서 윤곽이 그려집니다. 이 마커 안의 모든 내용은 호스트의 자바 가상 시스템처럼 스크립팅 언어 엔진에 의해 평가됩니다. JSP 사양은 표현과 선언을 포함하여 모든 일반적인 스크립트 요소를 지원합니다.

OC4J는 JSP 1.1 호환 번역기와 런타임 엔진을 제공합니다. 여기에는 몇 가지 중요한 기능이 있습니다.

· JSP 1.1을 위한 완벽한 지원: JSP Translator와 런타임은 모든 JSP Directives와 모든 코어/표준 JSP Tag를 위한 지원을 포함하여 JSP 1.1에 대해 완벽한 지원을 제공합니다.

· Simple, Body, Parameterized 및 Collaboration Tag: OC4J는 태그의 바디(body)가 한번만 평가되는 다음과 같은 단순 JSP 태그, 태그의 본문이 여러 번 평가되는 Body Tag(반복 루프에서와 같이), 태그가 파라미터를 수용하여 표시할 수 있는 Parameterized Tag 및 한 작업에 태그 두 개가 협력하도록 디자인된 Parameterized Tag의 특별한 종류인 Collaboration Tag를 지원합니다. 예를 들어 하나의 태그는 특정 값을 페이지 범위에 추가할 수 있으며, 이때 다른 태그는 계속 처리를 위해 이 값을 검색할 수 있습니다.

· JSP Caching Tag: JSP는 동적인 웹 페이지 생성 기술이기 때문에, 캐싱을 사용하면 JSP로 구축한 웹 사이트의 성능과 확장성이 향상될 수 있습니다. 오라클의 JSP Translator는 특정 JSP 태그를 공유 자바 캐쉬에서(인스턴스에 대한 추가 XSL_T 변환을 적용할 필요가 있는 경우) 또는 웹 캐쉬에서(최종 페이지가 클라이언트의 액세스를 위해 캐싱되는 경우) 캐쉬할 수 있는지 JSP 개발자가 표시할 수 있는 표준 구문을 제공합니다. 표준 JSP 태그 구문을 사용하여 특정 JSP 태그를 캐싱할 수 있는지의 여부를 표시함으로써 Oracle9*i*AS는 애플리케이션 개발자가 사용하는 캐싱 방법을 단순하게 만들며 캐싱할 수 있는 웹 페이지의 구성 요소를 보다 정확하게 정의할 수 있습니다(한 페이지 전체를 캐싱할 수 없는 경우에도). 표준 캐싱 확장자를 사용하면 Oracle9*i*AS Web Cache가 캐싱한 JSP 페이지를 사용할 수 있을 뿐 아니라 Akamai와 같은 인터넷 컨텐트 전달 네트워크를 통해 캐싱한 JSP 페이지도 사용할 수 있습니다.

· Mail, Search 및 기타 Tag: Oracle9*i*AS는 전자 우편을 송수신하고 파일(오라클의 Internet File System의 파일 포함)에 액세스하며 XML 결과를 JSP 페이지에 내장하고 웹 검색/질의를 실행할 수 있는 여러 가지 JSP 태그 라이브러리를 추가로 제공합니다.

· 완벽한 WAR 파일 기반 배치: 또한 Oracle9*i*AS는 1) JSP 페이지와 서블릿을 J2EE 표준 Web Application Archive(WAR) 파일로 패키징할 수 있는 툴과 2) 최종 WAR 파일을 선택하여 하나 이상의 Oracle9*i*AS 인스턴스에 배치하는 배치 툴을 제공합니다. WAR Deployment Tool은 클러스터 배치도 지원하여, 특정 아카이브를 클러스터 구성을 위해 정의한 모든 Oracle9*i*AS 인스턴스에 동시에 배치할 수 있습니다.

C. Enterprise JavaBeans(EJB)

Enterprise JavaBeans(EJB) 구성 요소는 개발자가 데이타베이스 액세스, 트랜잭션 지원, 보안, 캐싱 및 동시성과 같은 일반적인 사항에 대해 코드를 프로그래밍할 필요 없도록 비즈니스 로직을 캡슐화하도록 디자인되었습니다. EJB 사양에서 그러한 구성 요소는 EJB 컨테이너에서 취급합니다. 엔터프라이즈 빈은 인터페이스와 클래스로 구성됩니다. 클라어언트는 엔터프라이즈 빈의 홈과 원격 인터페이스를 통해 엔터프라이즈 빈 메소드에 액세스합니다. 홈 인터페이스는 엔터프라이즈 빈의 작성, 제거 및 위치 지정에 관한 메소드를 제공하며 원격 인터페이스는 비즈니스 메소드를 제공합니다. 컨테이너는 배치 시간으로 엔터프라이즈 빈의 비즈니스 메소드를 작성하고 제거하며 위치를 파악하고 호출하려는 클라이언트에 대한 액세스를 제공하기 위해 사용하는 인터페이스에서 클래스를 작성합니다. 엔터프라이즈 빈 클래스는 비즈니스 메소드, 작성 메소드 및 파인더 메소드를 위한 구현을 제공합니다. 빈이 자체의 지속성을 관리할 경우 라이프사이클에

대한 메소드도 제공합니다. 엔터프라이즈 빈에는 세션 빈 및 엔티티 빈의 두 가지 유형이 있습니다.

· 세션 빈: 세션 빈은 클라이언트와의 과도 대화(transient conversation)를 나타내며 데이타베이스 읽기와 쓰기를 실행할 수 있습니다. 세션 빈은 JDBC 호출 자체를 호출하거나 엔티티 빈을 사용하여 호출할 수 있습니다. 이 경우 세션 빈은 엔티티 빈에 대해 클라이언트가 됩니다. 세션 빈의 필드에는 대화의 상태가 포함되며 필드 자체는 과도 상태입니다. 서버나 클라이언트가 충돌할 경우 세션 빈은 손실될 수 있습니다. 세션 빈은 상태유지(stateful) 또는 무상태(stateless)일 수 있습니다.

· 상태유지 세션 빈: 상태유지 세션 빈에는 클라이언트를 대신하는 대화 상태가 수록됩니다. 대화형 상태는 세션 빈의 인스턴스 필드이자 세션 빈에서 취급할 수 있는 모든 오브젝트입니다. 상태유지 세션 빈은 지속성 데이타 스토어의 데이타를 나타내지는 않지만 클라이언트를 대신하여 데이타에 액세스하고 갱신할 수 있습니다.

· 무상태 세션 빈: 무상태 세션 빈에는 특정 클라이언트에 대한 상태 정보가 없습니다. 이 빈은 일반적으로 어떤 특정 상태로 유지하지 않는 서버측 행동을 제공합니다. 무상태 세션 빈은 시스템 자원을 많이 필요로 하지 않습니다. 일반 서비스를 제공하거나 저장된 데이타의 공유 뷰를 나타내는 비즈니스 오브젝트가 무상태 세션 빈의 좋은 예입니다.

· 엔티티 빈: 엔티티 빈은 데이타베이스의 데이타와 그 데이타에 작용되는 메소드를 나타냅니다. 종업원 정보 테이블을 위한 관계형 데이타베이스 맥락에서는 테이블의 각 행에는 하나의 빈이 있습니다. 엔티티 빈은 트랜잭션 형이며 데이타베이스에 데이타가 있는 한 존속합니다. 엔티티 빈은 컨테이너가 관리하는 지속성(Container Managed Persistence) 나 빈이 관리하는 지속성 (Bean Managed Consistency)을 지원할 수 있습니다.

· Container Managed Persistence(CMP): CMP를 사용하여 애플리케이션 개발자는 EJB 컨테이너가 Entity Bean과 지속적 스토어와의 대화를 투명하게 매핑하고 관리하기 때문에 Entity Bean을 지속적 데이타베이스 스토어에 매핑할 필요가 없습니다. 그 결과 엔티티 빈이 CMP를 사용함으로써 개발자는 데이타베이스 액세스에 JDBC 2.0 API가 필요 없으며 따라서 CMP의 사용이 단순하고 쉬워집니다. 그러나 애플리케이션 서버와 데이타베이스 간의 대화에 대한 애플리케이션 개발자의 제어가 제한되므로 성능 오버헤드가 어느 정도 필요할 수 있습니다.

· Beam Managed Persistence(BMP): 반대로 BMP는 지속적 스토어에서 엔터프라이즈 빈이 상태를 쓰고 읽는 방식을 최대한으로 제어하고자 하는 개발자가 사용할 수 있습니다. BMP는 애플리케이션 개발자가 데이타 로딩과 저장 및 런타임과 지속성 데이타베이스 스토리지 사이의 일관성 유지를 취급하기 위해 JDBC 2.0 API로 빈의 라이프사이클 메소드를 구현해야 하기 때문에 CMP보다 복잡합니다. BMP는 EJB의 완전한 제어를 모색하는 개발자가 사용해야 하며 또한 EJB가 비관계형 데이타베이스로 백업되는 경우 사용해야 합니다.

OC4J는 다음과 같은 기능을 갖는 JDK 기반 EJB Container를 제공합니다.

· EJB 1.1와 EJB 2.0의 중요 부분에 대한 완벽한 지원: EJB Container는 세션 빈과 엔티티 빈에 대한 완벽한 지원을 비롯해, EJB 1.1의 전체 지원과 Bean Managed Persistence(BMP) 및 Container

Managed Persistence(CMP)에 대한 완벽한 지원을 제공합니다. 또한 OC4J는 표준 O-R 매핑과 메시지 구동 빈과 같은 EJB 2.0의 중요한 부분(새로 발표할 J2EE 1.3 사양의 일부)을 구현합니다.

· 완벽한 Container Managed Persistence(CMP) 및 Bean Managed Persistence(BMP) 구현: Oracle9*i*AS는 자체의 오브젝트 관련 매핑(O-R)과 EJB 2.0 사양에 명시된 O-R 매핑을 모두 지원하는 Entity Beans에 대한 완벽한 CMP와 BMP를 제공합니다. OC4J의 Entity Beans 지원에는 다음과 같은 몇 가지 중요한 기능이 있습니다.

· 단순한 O-R 매핑: OC4J는 1:1, Many:1 및 1:Many 오브젝트 관련 매핑을 지원합니다. 엔티티 빈의 필드를 해당 데이타베이스 테이블에 자동으로 매핑하는 단순한 기능을 제공합니다. 또한 사용자는 OC4J를 사용하여 EJB 사이의 오브젝트 관계형 매핑을 지정함으로써 개발자는 어떤 작업도 할 필요 없이 EJB에서 1:1 Mapping을 사용할 수 있습니다(EJB 2.0 사양의 부분 기능입니다).

· 복잡한 O-R 매핑: 일반적인 문제는 매핑을 하기 위해 사용자 정의 코드를 작성하지 않고는 단순한 필드를 가진 단순한 필드 이외의 어떤 것도 데이타베이스 테이블에 매핑하기 어렵다는 것입니다. 그 결과 대부분의 엔티티 개발은 CMP를 사용하는 단순(1:1) 오브젝트 모델과 BMP를 사용하는 실질적(보다 복잡한) 오브젝트 모델, 이 두 개 범주로 분류됩니다. OC4J에는 복잡한 오브젝트 모델을 쉽게 데이타베이스 테이블에 매핑할 수 있는 O-R 매핑 시스템이 포함되어 있으므로 실질적 오브젝트 모델이 CMP를 사용할 수 있습니다. 특히 다음과 같은 유형의 필드를 엔티티 빈 안에서 매핑할 수 있습니다. 즉, 단순 오브젝트와 기본(예를 들어 INT, CHAR) 오브젝트(복합 오브젝트), 직렬화 오브젝트(직렬화하여 BLOB과 CLOB에 저장할 수 있는 복합 오브젝트), 엔티티 참조(다른 엔티티 빈에 대한 참조) 및 모음이 그들입니다. 또한 복잡한 O-R 매핑에서는 CMP 기능들이 오라클과 비 오라클 데이타베이스를 대상으로 할 수 있는, 자동으로 생성된 코드인 SQL을 포착하는 분리 계층을 제공합니다.

· 동적 EJB 부분 생성: 애플리케이션 개발자는 OC4J를 사용하면 ejbc, rmic 또는 다른 기능으로 EJB 부분을 클라이언트 애플리케이션으로 미리 컴파일할 필요가 없습니다. 그 대신 Oracle9*i*AS EJB 컨테이너가 필요에 따라 EJB를 생성하므로 경쟁사 제품보다 애플리케이션과 시스템 유지가 훨씬 쉽습니다.

· 전체 EAR 파일 기반 배치: OC4J는 1) WAR와 Enterprise JavaBeans를 표준 J2EE Compliant Enterprise Application Archive(EAR 파일)로 패키지하고 2) 배치 툴이 발생하는 .EAR 파일을 선택하여 하나 이상의 Oracle9*i*AS 인스턴스로 배치할 수 있는 툴을 제공합니다. EAR 배치 툴은 클러스터 배치도 지원하여, 특정 아카이브는 클러스터를 구성하도록 정의된 모든 Oracle9*i*AS 인스턴스에 동시에 배치할 수 있습니다.

· EJB 애플리케이션의 단순화되고 자동으로 이루어지는 배치: J2EE Application에는 두 가지 종류의 배치 기술자(descriptor) 또는 모듈 구성 파일이 있습니다. 모든 애플리케이션 서버가 지원하는 일반 J2EE 배치 파일과 업체 고유 파일, 즉 일반 파일은 애플리케이션 개발자나 구성 요소 어셈블러가 사용하며, 업체 고유 파일은 애플리케이션 배치자(deployer)가 사용합니다. OC4J는 다음과 같은 두 가지 방식으로 애플리케이션 서버 고유 배치 정보를 지원합니다.

· Auto-Deployment: 자동 배치의 경우 오라클 고유의 배치 정보는 J2EE EAR 파일이 서버에서 배치될 때

자동으로 생성됩니다.

· Simplified Configuration Customization: 또한 오라클 고유의 구성 정보는 애플리케이션 서버 고유 배치 및 구성 정보를 포착하는 XML 구성 파일의 단순한 세트를 수동으로 편집하면 사용자 정의할 수 있습니다. 여기에는 다음과 같은 설정이 포함됩니다. 즉, CMP용 자동 생성 및 자동 삭제 테이블, 보안 역할 매핑, JNDI Namespace 액세스, 세션 지속성 및 시간 초과 설정, 트랜잭션 재시도 설정, CMP 및 O-R 매핑, 버퍼링, 문자 세트, 지역성, 가상 디렉토리, 클러스터 구성, 세션 추적, 개발 및 디버깅 모드 설정 등입니다.

· Hot Deployment: 애플리케이션 개발자가 이미 배치된 EJB 모듈을 변경할 경우 개발자는 EJB를 다시 배치하거나 서버를 다시 시작할 필요가 없습니다. 사용자는 server.xml 구성 파일을 편집하고, 서버는 파일을 읽으며, 자동으로 변경 내용이 선택됩니다.

D. 자바 Database Connectivity Service (JDBS)

기본적으로 관계형 데이타베이스에 대한 휴대용 브리지로서 JDBS는 ODBS(Open Database Connectivity) 사양을 본떠 만들 것으로서 매우 단순하고 이해하기 쉽습니다. JDBS는 드라이버의 사용을 통해 데이타베이스를 프로그램 코드로부터 분리시킵니다. Oracle9*i*AS v1.0.2.2는 JDBC Driver에 자바에서 오라클 및 비 오라클 데이타베이스에 액세스할 수 있는 권한을 제공합니다.

· JDBC를 통한 오라클 데이타베이스 액세스: 오라클은 Oracle9*i*AS v1.0.2.2를 사용하여 Oracle8.0, 8*i* 및 9*i*Database에 액세스할 수 있는 권한을 다음과 같은 두 가지 JDBC 드라이버에 제공합니다.

· Oracle JDBC-OCI Driver: JDBC-OCI는 Oracle OCI 라이브러리를 사용하여 오라클 데이터베이스와 통신하는 Type 2 Oracle JDBC 드라이버입니다. 현재 이 드라이버는 클라이언트-서버 애플리케이션을 구축하는 개발자가 사용하며 기본 JDBC 드라이버도 Oracle9*i*AS 미들 티어에서 J2EE Application을 Oracle Database에 실행하는 J2EE 애플리케이션에서의 통신에 사용됩니다. 이 드라이버를 사용하려면 미들 티어에 오라클 클라이언트 라이브러리를 설치해야 합니다.

· Oracle Thin JDBC Driver: Oracle Thin JDBC는 완전히 자바로 구현되며 자바로 구현된 SQL*Net(Net8) 프로토콜을 사용하여 Oracle Database와 통신하는 2MB의 순수 자바(Type 4) JDBC 드라이버입니다. 애플리케이션 개발 관점에서 씬 JDBC 드라이버는 개발과 테스트 중 사용할 수 있습니다. 즉, 씬 드라이버의 순수 자바 호출 스택을 사용하면 디버깅을 철저하게 수행할 수 있습니다. 또한 Oracle Database와 직접 통신하는 자바 애플릿으로 다운로드할 수도 있습니다. 씬 JDBC 드라이버는 Oracle JDBC-OCI 드라이버와 100% 호환됩니다. 사용자가 변경해야 하는 유일한 항목은 Oracle DB와의 연결에 사용되는 연결 문자열입니다.

· 완전한 JDBC 2.0 지원: 오라클의 JDBC 드라이버는 JDBC 2.0과 완전히 호환되며 다음과 같은 기능을 제공합니다.

· 완벽한 데이타 타입 지원: BLOB, CLOB, Character Stream, Abstract Data Type, Collection과 같은 고급 데이타 타입 지원 및 Oracle9I Database Release로 Abstract Data Type with Inheritance 지원

· JDBC 2.0 Connection Pooling: 또한 JDBC 2.0 Connection Pooling 기능을 지원합니다.

· 향상된 기능: 또한 Oracle Database가 장애복구할 때 미들 티어가 장애복구 노드로 연결을 재지정될 수 있도록 해주는 Transparent Application Failover, 이동할 수 있는 결과 세트, 일괄 갱신, Unicode 지원 및 기타 몇 가지 고급 기능을 제공합니다.

· Oracle8.0, 8*i*, 9*i* 지원: Oracle9*i*AS v.1.0.2.2 JDBC Driver는 Oracle8.0, 8*i* 및 9*i* Database Release 1로 인증된 제품입니다.

· Merant JDBC Driver: OC4J에서 비 오라클 데이터베이스에 액세스하기 위해 오라클은 Merant(오라클 파트너)의 Type 4 JDBC Driver를 권장합니다. Merant는 Oracle9*i*AS에서 Informix, Sybase, Mocrosoft SQL-Server 및 IBM DB/2 Database에 액세스할 수 있는 JDBC Driver를 제공합니다.

E. 자바 Naming and Directory Interface (JNDI)

JNDI는 이름 지정 및 디렉토리 서비스에 대한 표준 인터페이스입니다. J2EE 애플리케이션은 JNDI를 사용하여 다른 분산 오브젝트를 찾습니다. JNDI Interface는 두 부분으로 구성됩니다. 즉, 이름 지정과 디렉토리 서비스에 액세스하기 위해 애플리케이션 프로그램이 사용하는 애플리케이션 수준 인터페이스와 이름 지정과 디렉토리 서비스의 제공업체를 첨부하기 위한 서비스 제공업체 인터페이스가 그것들입니다.

Oracle9*i*AS v.1.0.2.2는 완벽한 JNDI 1.2 구현을 제공합니다. Servlet과 Enterprise JavaBeans는 표준 JNDI 프로그래밍 인터페이스를 사용하여 이름에 액세스합니다. JNDI 서비스 제공업체는 XML 기반 파일 시스템에서 구현됩니다. Oracle9*i*AS Release 2.0은 파일 시스템 기반 JNDI를 유지하는 한편 LDAP 사용을 위한 성능을 JNDI 서비스 제공업체의 대안으로도 제공합니다.

F. 자바 Transaction API (JTA)

애플리케이션 개발자는 J2EE 트랜잭션 모델을 사용하여 배치 시간에 단일 트랜잭션을 구성하는 메소드 사이의 관계를 지정할 수 있습니다. 따라서 어느 한 트랜잭션의 모든 메소드는 단일 단위로 취급됩니다. J2EE Standard에는 트랜잭션이 모두 완료되거나 제거되어야 하는 일련의 단계이기 때문에 그러한 지원이 필요합니다. 예를 들어 첫번째 계정에서 출금하여 두 번째 계정에 입금함으로써 돈을 하나의 계정에서 다른 계정으로 이동하는 엔터프라이즈 빈에 몇 가지 메소드가 있을 수 있습니다. 전체 운영을 하나의 단위로 취급할 때 출금 후와 입금 전 장애가 발생하면 출금이 롤백합니다. 어셈블리 중 애플리케이션 구성 요소에 트랜젝션 속성이 지정됩니다. 따라서 메소드를 애플리케이션 구성 요소 전체에서 트랜잭션으로 그룹화할 수 있으므로 J2EE 애플리케이션 안에서 애플리케이션 구성 요소를 쉽게 변경할 수 있으며 코드를 변경하거나 재 컴파일할 필요 없이 트랜잭션 속성을 재할당할 수 있습니다.

자바 Transaction API(JTA)와 자바 Transaction Service(JTS)는 J2EE, 특히 EJB와 JDBC 2.0에서의 트랜잭션 지원을 위한 기초를 형성합니다. JTS는 기본적으로 자바를 Object Management Group(OMG) Object Transaction Service로 매핑하는 트랜잭션 관리를 위한 로우 레벨 API입니다. JTA는 다음과 같은 두 부분으로 구성되는 하이 레벨 API입니다.

· Transaction Interface: Transaction Interface는 트랜잭션 분할을 허용하며, 전역 트랜잭션에 의해 구속되는 분산 구성 요소로 작업을 수행할 수 있습니다. 또한, 트랜잭션을 구성하기 위한 운영 그룹을 표시하는 방법입니다.

· XA Resource Interface: XA Resource Interface는 분산 트랜잭션을 취급할 수 있는 X/Open/XA 인터페이스를 기반으로 합니다. 이들은 Two Phase Commit 트랜잭션이라고 하는 경우도 있으며 데이타베이스나 큐와 같이, 둘 이상의 자원에서 이루어지는 트랜잭션의 조정을 수반합니다. 개발자들이 J2EE를 사용하면 JTA와의 명백한 트랜잭션을 프로그래밍할 필요가 없는데, 컨테이너에 의해 취급되고 애플리케이션 배치 기술자에 의해 구성되는 JDBC와 EJB API를 통해 작업이 이루어지기 때문입니다. 애플리케이션 개발자는 구현보다는 트랜잭션의 디자인에 집중할 수 있습니다.

OC4J는 JTA 1.0.1 사양의 완벽한 구현을 제공합니다. Oracle9*i*AS Release 2.0은 오라클과 비 오라클 XA 호환성 Resource Manager 전체에 확장성이 뛰어난 1PC 및 2PC(Two Phase Commit)를 제공하는 오라클 데이터베이스의 Commit Coordinator에 기초한 미들 티어 Distributed Commit Coordinator에 JTA 기능을 통합합니다.

G. 자바 Messaging Service (JMS)

JMS는 자바 프로그램 사이에서 이루어지는 메시지 교환을 지원하는 J2EE 메커니즘입니다. 이것이 자바가 비동기 통신을 지원하는 방식입니다. 즉, 송신자와 수신자가 상대를 알 필요가 없으므로 독립적으로 운영할 수 있습니다. JMS는 다음과 같은 두 가지 메시징 모델을 지원합니다.

· 지점간(point to point): 메시지 큐에 기초합니다. 메시지 생산자가 메시지를 큐로 보냅니다. 메시지 소비자가 큐에 연결하여 메시지를 들을 수 있습니다. 메시지가 큐에 도착하면 소비자는 큐에서 메시지를 선택하여 그에 대한 응답을 합니다. 메시지는 큐 하나에만 보내 소비자 한명만 사용할 수 있습니다. 소비자는 원하는 정확한 메시지 유형 지정을 위해 메시지를 필터링할 수 있습니다.

· 발행 및 인용(publish and subscribe): 생산자가 메시지를 주제로 보내며 그러한 주제에 대해 등록된 모든 소비자가 메시지를 검색할 수 있는 모델입니다. 이 경우 하나의 메시지를 여러 소비자가 받을 수 있습니다.

OC4J는 JMS 1.0 사양의 완벽한 구현을 제공합니다. Oracle9*i*AS Release 2.0은 JMS를 오라클의 Advanced Queuing Messaging Service에 통합합니다.

H. 자바 보안 서비스

보안과 트랜잭션 관리와 같은 애플리케이션 작업은 웹과 엔터프라이즈 빈 구성 요소에서 배치 시간으로 구성할 수 있습니다. 이 기능은 어셈블리에 따라 다를 수 있는 구성 설정값에서 애플리케이션 로직의 결합을 해제합니다. J2EE 보안 모델을 사용하면 웹이나 엔터프라이즈 빈 구성 요소를 구성할 수 있으므로 허가된 사용자만 시스템 자원에 액세스할 수 있습니다. 예를 들어 사용자 이름과 암호를 묻도록 구성 요소를 구성할 수 있습니다. Enterprise Bean 구성 요소는 특정 그룹의 담당자만 특정 메소드 종류를 호출할 수 있도록 구성할 수 있습니다. 또는 일부 메소드는 모든 사람이 액세스할 수 있는 반면, 다른 메소드는 조직의 특권이

있는 사람만 액세스할 수 있도록 서블릿 구성 요소를 구성할 수 있습니다. 같은 서블릿 구성 요소라도 모든 사람이 모든 메소드를 사용할 수 있거나, 모든 메소드가 소수 제한된 사람만 사용할 수 있는 등 다른 환경으로 구성될 수 있습니다.

OC4J에는 서버에서 실행되는 구성요소의 사용을 정밀하게 제어할 수 있는 아주 강력한 Access Control List(ACL) 메커니즘이 있습니다. 자바 메소드 수준까지 어떤 사용자나 사용자 그룹이 어떤 내용을 실행할 수 있는지 정의할 수 있습니다. 이 ACL 메커니즘은 EJB를 제외하고 OC4J에서 실행되는 모든 내용을 취급합니다. 여기에는 EJB 사양에 정의된 자체의 액세스 제어 메커니즘이 있습니다. 관리자는 보안 영역을 사용하여 기존의 허가 또는 승인 시스템에서 ACL로 정보를 임포트할 수 있습니다. 따라서 NT 보안 시스템, LDAP 시스템, UNIX 암호 파일 또는 데이타베이스에서 정보를 임포트할 수 있습니다. Oracle9*i*AS v.1.0.2.2는 보안에 관한 한 상당히 완벽합니다. 여기에는 1) Secure Socket Layer(SSL) 버전, 2) RSA Encryption 및 3) Support for X.509 증명서 버전 3을 위한 모든 클래스가 포함됩니다. 또 하나 관련된 보안 주제는 방화벽 터널링의 기능입니다. Oracle9*i*AS v.1.0.2.2는 HTTP와 HTTPS 터널링을 통해 방화벽과 프록시를 통과할 수 있는 성능을 제공합니다. J2EE 보안 기능은 Oracle9*i*AS Release 2.0을 사용하여 보안을 위한 단일 서명(sign on)과 단일 스테이션 관리에 대한 Oracle Login Server를 지원합니다.

I. RMI와 HTTP 터널링 서비스

배치되는 J2EE Application은 HTTP Listener가 배치되는 Web Server Tier 및 JSP와 Servlet이 배치되는 Web Presentation Tier, EJB로 정의된 Business Logic이 배치되는 EJB Tier 등 두 세가지 티어로 구분됩니다. 규모가 작은 웹 사이트는 이 세 가지 티어를 하나의 물리적 미들 티어로 결합합니다. 규모가 큰 웹 사이트에서는 이들 티어를 보안, 확장성 및 로드 밸런싱 목적을 위해 두 세 개 별도의 물리적 티어로 분리합니다. Oracle9*i*AS는 이러한 구조적 문제를 고려하여 다음과 같은 요구를 충족하도록 디자인되었습니다.

- 웹 서버 대 JSP/서블릿 엔진 연결성: 웹 서버는 Apache ajp 프로토콜이나 HTTP를 사용하여 요청을 JSP/서블릿 엔진으로 보낼 수 있습니다. 그 결과 웹 서버는 방화벽 밖에도 배치할 수 있으며 요청을 방화벽 뒤에 위치한 서블릿 엔진으로 보낼 수 있습니다.
- 자바/서블릿 대 EJB 및 EJB 대 EJB 연결성: Presentation Logic Tier에서 Business Logic Tier까지 및 EJB들 간의 통신은 EJB에 액세스하는 클라이언트나 웹 티어 프로그램에 EJB 티어 안의 서비스에 직접 액세스할 수 있는 권한을 부여하는 표준 RMI를 사용하여 이루어집니다. 이들 서비스에는 EJB를 검색하여 참조하는 JNDI, 비동기 메시지를 송수신하기 위한 Java Messaging Service(JMS) 및 관계형 데이타베이스 액세스를 위한 JDBC가 포함됩니다.
- HTTP와 HTTP-S 터널링: 또한 Oracle9*i*AS는 HTTP와 HTTP-S 프로토콜을 통해 RMI를 터널링할 수 있는 기능을 지원합니다. HTTP/HTTP-S 터널링을 통한 RMI는 Oracle9*i*AS와 통신할 필요가 있지만 유일한 옵션이 HTTP 프로토콜을 사용하는 것일 경우 자바 기반 클라이언트에 사용됩니다. 일반적으로 HTTP 터널링은 자바 클라이언트와 Oracle9*i*AS 사이의 상태 유지(stateful) 소켓 연결을 시뮬레이션하고 보안 방화벽의 HTTP 포트를 통해 이 소켓 연결을 터널링하는 방법을 제공합니다. HTTP는 무상태(stateless) 프로토콜이지만 Oracle9*i*AS가 터널링 기능을 제공하여 연결을 정규 상태유지(stateful) RMI

Connection처럼 보이게 합니다. 클라이언트는 HTTP 프로토콜에서 요청을 한 다음 서버에서 응답을 받을 수만 있습니다. 서버는 스스로 클라이언트와 통신하지 않으며 프로토콜은 무상태(stateless)입니다. 즉, 연속적인 양방향 연결이 불가능합니다. Oracle9*i*AS의 HTTP 터널링은 HTTP 프로토콜을 통해 RMI Connection을 시뮬레이션함으로써 이러한 제약을 극복합니다.

그 결과 Oracle9*i*AS v.1.0.2.2의 다양한 J2EE 구성 요소를 단일 물리적 티어에 배치하거나(보통 성능의 최적화를 위해) 별도의 물리적 티어에 배치할 수 있습니다(높은 가용성을 위한 연결 경로 재지정과 같은 이중화를 위해)

J. 요약

요약하면, OC4J의 기능을 갖는 Oracle9*i*AS v.1.0.2.2는 JSP Translator, 자바 서블릿 엔진 및 Enterprise JavaBeans(EJB) 컨테이너가 포함된 완벽한 J2EE 컨테이너를 제공하는 J2EE에 대한 오라클의 지원이 상당히 진전했음을 의미합니다.

J2EE에 필요한 표준 인터페이스	9*i*AS v.1.0.2.2 지원
자바 Server Page(JSP)	1.1
Servelet2.2	
Enterprise JavaBeans(EJB)	1.1
자바 Translation API(JTA)	1.0.1
자바 Managing Service(JMS)	1.0
자바 Naming and Directory Interface(JNDI)	1.2
자바 Mail1.1.2자바 Database Connectivity(JDBC)	2.0

그림: Oracle9*i*AS v.1.0.2.2 J2EE 지원

OC4J는 Enterprise JavaBeans(EJB) 1.1, Servlet 2.2, 자바 Server Page(JSP) 1.1, JTA 1.0.1, JNDI 1.2, JMS 1.0, JDBC 2.0 및 자바Mail 1.1.2와 같은 J2EE 1.2 API를 모두 완벽하게 지원합니다. Oracle9*i*AS의 최신 릴리스는 인증된 J2EE입니다.

3. 활용성, 성능, 확장성 및 가용성

이제 OC4J의 성능을 모두 이해한 것으로 간주하고 환경의 사용성, 성능, 확장성 및 가용성을 검토하겠습니다.

A. OC4J의 Footprint/활용성

Oracle9*i*AS v.1.0.2.2는 J2EE Container/Server의 디스크와 메모리 용량이 상당히 작아도 사용할 수 있습니다. 그 결과 J2EE Environment 설치와 실행에 필요한 하드웨어 원가가 상당히 절감되며 J2EE Application 개발과 배치의 활용성이 강화됩니다. Oracle9*i*AS v.1.0.2.2를 사용하여 J2EE Application을 아주 쉽게 개발하고 배치할 수 있는 중요한 요인 세 가지는 1) 경량 제품 Footprint, 2) 애플리케이션의 단

순화된 배치와 구성, 3) 단순화된 애플리케이션 디버깅입니다.

· 경량 제품 Footprint: OC4J는 JDK 자바 Virtual Machine에서 완벽하게 배치되므로, 이전의 Oracle9*i*AS 자바 환경이나 경쟁사의 자바 애플리케이션 서버보다 디스크와 메모리의 footprint가 크게 줄어들었습니다. 아래 차트에 Oracle9*i*AS v.1.0.2.2의 footprint 요구사항이 요약되어 있습니다.

영역	Oracle9*i*AS v.1.0.2.2	WebLogic 5.0	IBM WebSphere
다운로드 파일 크기	10MB	31MB	45MB
디스크 Footprint	15MB	45MB	66MB
최소 메모리 Footprint	20MB	256MB5	12MB

그림: OC4J Footprint 및 경쟁사 제품의 Footprint

· 서버의 단순화된 배치 및 구성: OC4J는 설정과 구성이 매우 쉽습니다. 표준 zip 파일을 설치하거나 다운로드하면 설치와 구성이 완료됩니다. zip 파일을 다운로드한 후 이 파일의 압축을 풀고 JAR(자바 ARchive) 파일을 지정된 루트 디렉토리에 복사하기만 하면 설치가 끝납니다. 단 하나의 명령으로 J2EE 1.2 애플리케이션 서버를 연결하여 실행할 수 있습니다. 또한 비교적 복잡한 시스템 구조에도 자바 Container의 구성이 아주 쉽습니다. J2EE Container 자체의 구성에 필요한 모든 정보는 6개의 XML 파일로서 포착됩니다. 그 결과 XML 속성 파일을 편집하기만 하면 컨테이너에서 배치되는 J2EE Container와 J2EE Application의 구성을 모두 변경할 수 있습니다. 6개의 XML 구성 파일은 다음과 같습니다.

- server.xml: server.xml 파일에는 라이브러리 경로 이름, 전역 애플리케이션 이름, 서버가 허용하는 최대 연결 숫자, 로깅 설정, 자바 컴파일러 설정, 클러스터 ID, 트랜잭션 시간 초과 및 SMTP 호스트와 같은 애플리케이션 서버 자체 구성 정보가 수록됩니다. 또한 server.xml 파일에는 다른 구성 파일에 대한 참조도 수록되며 자체 애플리케이션을 추가할 위치에 관한 정보도 포함되어 있습니다.
- web-site.xml: web-site.xml 파일에는 특히 사이트의 호스트 이름과 가상 호스트 설정값과 같은 웹 사이트 구성, 액세스 로그 형식, SSL 구성 정보 및 사용자 웹 애플리케이션의 설정값이 수록됩니다.
- principals.xml: principals.xml 파일은 보안 목적을 위한 사용자와 그룹 구성 정보의 위치입니다.
- data-source.xml: data-source.xml 파일에는 사용되는 데이타 원본의 구성 정보가 수록됩니다. 데이타 원본에는 JDBC 드라이버, JDBC URL, 데이타 원본 바인딩에 사용되는 JNDI 경로, 사용할 데이타베이스 스키마, 대화성 시간 초과 및 데이타베이스가 허용하는 최대 연결 숫자와 같은 설정값이 있습니다. 데이타베이스 스키마는 Container Managed Persistence에 사용되는 자동 생성 SQL을 여러 가지 데이타베이스 시스템에서 운영할 수 있는 목적에 사용됩니다.
- rmi.xml 및 jms.xml: rmi.xml 및 jms.xml 파일은 JMS와 RMI 시스템의 구성 정보를 설정하여 RMI(JMS 포함) 서버가 설정값을 바인드하고 로깅하는 호스트 이름/포트를 사용자가 지정할 수 있습니다. 또한 jms.xml 파일을 사용하면 JNDI 트리에 구속되는 큐와 주제를 지정할 수 있으며 RMI 서버를 사용하면 통신할 원격 서버와 사용할 클러스터링 설정값을 지정할 수 있습니다.

· 단순화된 애플리케이션 배치 및 구성: 앞의 절에서 설명한 것처럼 OC4J는 Oracle9*i*AS 자체에서 배치되는 J2EE 애플리케이션을 구성하도록 편집할 수 있는 XML 구성 파일을 제공합니다. Oracle9*i*AS는 J2EE Application의 패키지와 배치하기 위한 다음과 같은 많은 툴을 제공합니다.

- 어셈블리 툴: Oracle9*i*AS는 J2EE Application 구성과 패키지를 위한 여러 가지 어셈블리 툴을 제공합니다. 이들 툴의 결과는 J2EE 표준을 준수하며 Oracle9*i*AS J2EE Container에만 해당되는 것은 아닙니다. 어셈블리 툴에는 1) JSP, Servlet, Tag Library 및 Static Content를 WAR 파일에 어셈블리하기 위한 WAR File Assembly Tool, 2) EJB Home, Remote Interface, Deployment Descriptor 및 EJB 자체를 표준 JAR 파일에 패키지하는 EJB Assembler, 3) WAR File과 EJB JAR를 표준 EAR 파일에 어셈블리하는 EAR File Assembly Tool 및 4) JSP Tab Library를 표준 JAR 파일에 어셈블리하는 Tablibrary Assembly Tool이 포함됩니다.

- 관리 툴: Oracle9*i*AS는 또한 다음과 같은 두 가지 관리 기능을 제공하여 OC4J를 구성하고 감시하며 관리합니다. 즉, 1) 명령 프롬프트에서 로컬로 또는 원격으로 관리 작업을 수행하도록 지원하는 Command Line Tool, 2) 하나 이상의 로컬 또는 원격 J2EE 컨테이너 전체에서 단일 관리 점을 제공하는 자바에 내장된 Graphical Management Console이 그것들입니다. 그래픽 콘솔은 Oracle Enterprise Manager와 통합됩니다.

· 단순한 애플리케이션 디버깅: 최근 애플리케이션 서버는 애플리케이션 개발자의 작업 상당 부분을 자동화하여 개발자의 효율이 증대되지만, 개발자가 기본적인 서버 환경에서 발생하는 오류를 분리하기란 점점 어려워지고 있습니다. 고급 자바 디버깅 툴조차 문제가 발생하는 위치는 알려주지만 문제 발생 원인은 알려주지 못합니다. 이러한 전체 프로세스를 단순하게 만들기 위해 OC4J는 오류와 이벤트 정보를 앞으로도 계속 참조하고 분석하기 위해 이벤트를 저장할 수 있는 로그 파일로 출력합니다. OC4J는 다음과 같은 5개의 서로 다른 로그 파일을 사용합니다.

- Web Access Log: Web Access Log는 웹 사이트나 인터넷 애플리케이션에서 발생하는 사용자/방문자 활동의 결정에 사용됩니다.

- Application log: 오류 진단에 가장 중요한 로그 파일입니다. 애플리케이션마다 애플리케이션 로그 파일이 하나씩 있으며 특정 애플리케이션에 관련된 모든 이벤트가 포함됩니다. 대표적인 애플리케이션 로그 파일은 다음과 같습니다.

```
2001-03-11 17:01 Started
2001-03-11 17:01 Auto-deploying shop-ejb… done
2001-03-11 17:01 shop-web: 1.3.3 Started
2001-03-11 17:01 shop-web: Servlet error
자바.lang.SecurityException: JSPFactory already set at
자바x.servlet.jsp.JSPFactory.setDefaultFactory(JSPFactory.자바)
at /shop/listProducts.jsp._jspService
(/shop/listProducts.jsp.자바:47)
```

2001-03-11 17:03 Stopped

로그 파일에는 오류를 던지는 애플리케이션의 실행 중 개발자가 특정 지점을 쉽게 파악하는 데 도움이 되는 정보가 수록됩니다.

- Server Log: Server Log에는 특정 애플리케이션이나 부속 시스템과 연관되지 않은 모든 이벤트가 포함되며, 서버 시작, 서버 종료 및 내부 애플리케이션 서버 오류의 로그가 포함됩니다.
- RMI 및 JMS Log: RMI 및 JMS Log는 RMI와 JMS 이벤트와 오류를 각각 포착하고 로그합니다. 이들 로그에서 가장 흔한 종류의 오류는 오브젝트의 마샬링(marshaling) 실패와 관련이 있습니다.

모든 로그 파일에는 공통 파일 형식이 있으므로 애플리케이션 개발자가 자동화 스크립트와 툴을 사용하여 오류를 파악하고 분석할 수 있습니다. 마지막으로 OC4J는 자바에서 자동으로 구현되기 때문에 속성을 속성 파일에 설정하기만 하면 몇 가지 디버깅 모드로 설정할 수 있습니다. 즉, 표준 자바 프로파일링과 디버깅 툴을 사용하여 서버 자체를 디버깅할 수 있습니다.

B. OC4J 성능

OC4J는 업계에서 속도가 가장 빠른 자바 애플리케이션 서버입니다. 오라클은 다음과 같은 별도의 백서를 제공합니다.

· 비교 가능한 서블릿 엔진 벤치마크: OC4J의 Servlet Engine을 경쟁사 애플리케이션 서버의 Servlet Engine과 비교한 벤치마크 번호

· 비교 가능한 J2EE 컨테이너 벤치마크: Servlet과 EJB를 모두 포함하는 OC4J를 경쟁사 애플리케이션 서버의 J2EE 컨테이너와 비교하는 벤치마크 번호. 오라클은 ecPerf Specification Standards Process에도 적극적으로 참여하고 있습니다. ecPerf는 J2EE Container의 성능을 비교하기 위한 표준 메커니즘과 벤치마크로서 논의되고 있습니다.

C. OC4J 확장성

J2EE를 사용하여 구축한 웹 사이트나 인터넷 애플리케이션의 성능에 관련된 핵심은 동시 사용자가 시스템 증가에 로딩할 때 J2EE Containder가 성능을 수행하는 정도에 있습니다. 이것을 보통 J2EE 컨테이너의 확장성이라고 합니다. OC4J에 관련된 확장성 이슈는 다음 네 가지 영역으로 구분됩니다.

· 하드웨어 확장성: OC4J가 다양한 하드웨어 플랫폼을 활용하여 확장성을 향상시키는 방법

· 자원 풀링과 관리: Oracle9*i*AS가 스레드, 메모리 및 데이타 연결과 같은 희소 자원을 최적화하여 확장성을 향상시키는 방법

· 클러스터 지원: 단일 가상 서버나 클러스터로 클라이언트의 요청을 서비스할 수 있도록 서버를 공유 상태로 구성하는 방법

· 로드 밸런싱: Oracle9*i*AS가 단일 가상 서버로서 효율적으로 운영되는 다양한 J2EE Container에 클라이언트의 요청을 분배하는 방법

하드웨어 확장성

시스템의 확장성은 일반적으로 시스템 증가에 대한 로드로서 복수의 동시 클라이언트 요청을 처리하는 시스템의 능력을 제한하는 병목현상에 의해 제한됩니다. 하드웨어 관점에서 두 가지 대표적인 확장성 병목현상은 CPU 제약(프로세서가 완전히 소진되기 때문에 하드웨어에 제약 초래)과 메모리 제약(하드웨어 시스템이 메모리 구속을 받기 때문에 제약 초래)입니다. 관리자는 이러한 문제를 해결하기 위해 보통 애플리케이션 서버를 높은 수준의 하드웨어 구성으로 이전하거나(시스템이 CPU로 제한될 경우) 메모리를 추가합니다(시스템이 메모리로 제한될 경우). Oracle9*i*AS를 사용하면, 애플리케이션을 전혀 변경하지 않고도 기본 하드웨어 구성이 업그레이드되므로 개발자는 투명하게 확장될 수 있는 J2EE 애플리케이션을 작성할 수 있습니다.

- 광범위한 OS/하드웨어 플랫폼의 가용성: OC4J는 로우 엔드 유니프로세서 시스템에서 하이 엔드 SMP 클러스터와 Solaris, HP-UX, AIX, Tru64, Windows NT 및 Linux와 같은 모든 주요 운영 체제에 이르기까지 광범위한 하드웨어 플랫폼에서 사용할 수 있습니다.
- 단일 노드 또는 복수 노드 클러스터: 또한 OC4J는 사용되는 하드웨어 플랫폼이나 OS와는 관계 없이 클러스터 지원을 제공합니다(클러스터 성능에 특정 SMP 하드웨어 구성이 필요하지 않은 경우). 따라서 OC4J 개발자와 관리자는 OS/하드웨어 플랫폼에 얽매이지 않고도 이러한 기능을 활용할 수 있습니다.

자원 풀링 및 관리

Oracle9*i*AS는 스레드, 메모리, EJB 인스턴스 및 데이타베이스와 같은 희소 자원을 최적화하여 단일 상자에서 지원할 수 있는 동시 J2EE 사용자의 수를 증가시킵니다. 확장성 관점에서, 희소 자원은 일반적으로 자원의 부족과는 대조적으로, 자원 취득의 어려움을 관찰함으로써 분류됩니다. 애플리케이션은 실행을 위해 자원에 액세스해야 하며 이러한 액세스는 애플리케이션의 로직에 의해 결정됩니다. OC4J를 사용하면, 수요가 늘 있는 자원을 관리하여, 동시에 실행되는 모든 애플리케이션 중 연속적이고 최적의 상태로 재사용될 수 있습니다. Oracle9*i*AS v.1.0.2.2는 다음과 같은 몇 가지 핵심 기능을 사용하여 이러한 자원의 할당 프로세스를 투명하고 효율적으로 관리합니다.

- 자원 풀의 제공
- 큐잉(queuing) 및 대기를 위해 고급 일정 계획/할당 알고리즘 채택
- 경우에 따라 자원을 공급 및 공급 거부

아래의 그림은 Oracle9*i*AS가 향상된 확장성을 위해 희소 자원의 사용을 취급하는 방법이 개념적으로 잘 보여주고 있습니다.

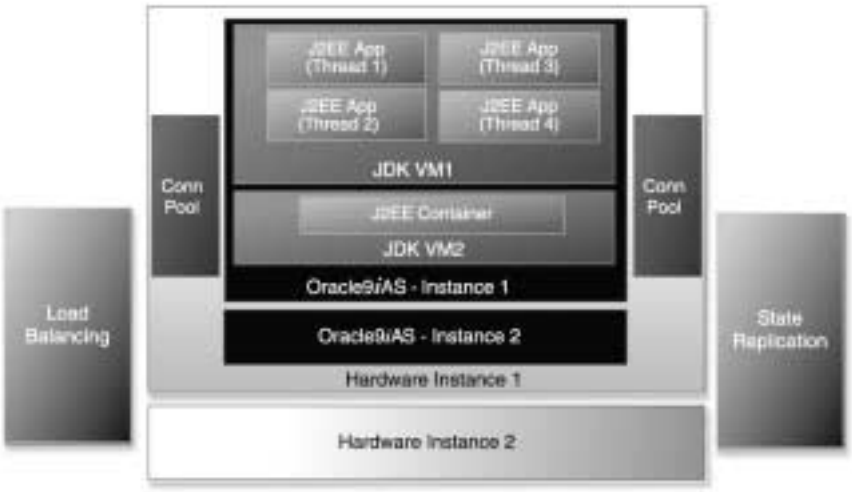


그림: Oracle9*i*AS v.1.0.2.2 확장성 구조

· 스레딩 모델: OC4J의 스레딩 모델은 여러 애플리케이션이 자원을 효과적으로 공유하도록 디자인되었습니다. 특히 OC4J는 J2EE Container의 계층을 다음과 같이 생성하고 관리합니다.

단일 하드웨어 인스턴스(상자)를 복수의 Oracle9*i*AS 인스턴스에서 실행할 수 있습니다.

- 단일 Oracle9*i*AS 인스턴스는 복수 자바 Virtual Machine(JDK)을 실행할 수 있습니다.
- 단일 자바 Virtual Machine(JDK)은 복수 스레드를 시작하고 유지할 수 있습니다.
- 각 스레드는 단일 J2EE Application 모듈(JSP, 서블릿 또는 EJB)을 실행할 수 있습니다.

또한 OC4J는 스레드 풀을 미리 시작하고 유지하여 요청을 처리합니다. J2EE에서 특정 요청을 받으면 휴지 스레드가 선택되며 특정 J2EE 모듈이 로딩되어 그 스레드에서 실행됩니다. J2EE 모듈이 그 요청의 처리를 완료하면 스레드는 해제되어 풀로 되돌아갑니다.

· 내향 요청 풀링(연결 풀링): 또한 Oracle9*i*AS는 소켓 연결의 풀을 유지하여 J2EE 클라이언트, Web Server, Load Balancer 또는 다른 J2EE Application 모듈의 요청을 취급합니다. OC4J는 특정 연결 요청을 수신하면 그 요청을 처리할 휴지 소켓을 선택합니다. 요청을 요청–응답 방식으로 처리할 수 있는 경우(무상태의 경우 HTTP 1.0 방식처럼) 응답이 소켓으로 작성되어 되돌아오면 소켓을 재사용합니다. 소켓을 계속 사용해야 할 경우(HTTP 1.1이나 클러스터링 목적을 위해) J2EE 컨테이너는 해제될 때까지 또는 시간 초과가 발생할 때까지 복수 요청에 대해 소켓을 유지합니다.

· 외향 요청 풀링(데이타베이스 연결 풀링): 데이타베이스 연결의 관리를 위해 자원 풀링을 나타내는 것이 가장 보편적인 경우입니다. 데이타베이스 연결 단계 중 보안, 전역화, 프로토콜 버전과 같은 몇 가지 문맥 파라미터를 구축해야 합니다. 이러한 구축에 따라 이 프로세스에 많은 비용이 소요됩니다. 이러한 문제를 해결하기 위해 OC4J는 데이타베이스에 대한 JDBC 연결의 풀을 미리 시작하여 유지합니다. 특정 요청의 경우 OC4J는 휴지 JDBC 연결을 풀에서 선택하여 재사용합니다. 이 프로세스가 완료되면 연결은 다른 클라이언트가 재사용할 수 있도록 풀로 반환됩니다. 연결이 이미 이루어졌고 클라이언트가 요구하는

즉시 사용할 수 있으므로 연결 풀링을 통해 성능이 상당히 증가하는 경우가 많습니다. 또한 복수 클라이언트가 연결 풀을 사용하여 적은 수의 데이타베이스 연결로 다중화되기 때문에 미들 티어와 데이타베이스 확장성도 개선됩니다(각 활성 클라이언트는 전용 데이타베이스 연결을 유지하기 않기 때문에 훨씬 효율적으로 사용됩니다. 대신 자원을 필요할 때만 사용할 수 있으므로 자원 낭비를 줄일 수 있습니다. 사실 데이타베이스 사용은 드물게 간헐적으로 발생하기 때문입니다).

로드 밸런싱

로드 분산 또는 로드 밸런싱은 기본적으로 1) 클라이언트 요청을 효율적으로 처리하고 2) 복수 Oracle9*i*AS 서버가 단일 가상 서버로 운영되는 것처럼 클라이언트에게 보이도록 클라이언트의 요청을 OC4J의 복수 인스턴스에 분산시키는 방법을 의미합니다. 로드 밸런싱을 효율적으로 수행하면 시스템 사이의 확장성이 극대화되며 처리 자원의 사용을 최적화할 수 있습니다. OC4J는 단일 노드와 복수 노드 배치의 노드 사이의 스레드와 프로세스에서 매우 정교한 로드 밸런스를 지원합니다.

- HTTP 서버에서의 로드 밸런싱: 웹 서버는 단순하면서도 효율적인 메커니즘을 사용하여 서버의 단일 인스턴스 안의 HTTP 서버 프로세스 사이에서 로드 밸런싱을 유지합니다. 마스터 HTTP 서버는 클라이언트 요청 자체는 처리하지 않지만 뮤텍스를 사용하여 공유된 소켓에서 HTTP 요청을 차례로 수용하는 자식 프로세스 그룹을 생성하고 감시합니다. 자식 프로세스가 요청을 받은 후 처리하기 전 다른 자식 프로세스가 취득한 뮤텍스를 해제합니다. 그 결과 소켓에 대한 액세스는 직렬화되지만 자식 프로세스는 요청을 동시에 취급할 수 있습니다. 또한 오라클 HTTP 서버는 하위 프로세스가 DNS 라운드 로빈이나 전용 하드웨어 로드 밸런서와 같은 다양한 기술을 사용하여 별도의 호스트 인스턴스에 로드의 밸런스를 유지할 수 있는 복수의 노드에서 실행할 수 있습니다(아래 설명 참조).
- OC4J에서의 로드 밸런싱: 서블릿이나 EJB 컨테이너 인스턴스는 단일 노드 또는 복수 노드의 인스턴스에서 이루어지는 요청의 로드를 골고루 분배합니다. 이들은 라운드 로빈, 임의, 일치 등과 같은 다양한 로드 밸런싱 알고리즘을 사용하여 시스템 관리자가 다양한 컨테이너 인스턴스에 가중치를 제공하는 가중 알고리즘에 기초한 여러 가지 컨테이너에 요청을 분배하는 가중 라운드 로빈 알고리즘도 사용합니다. 이와 같은 방식으로 보다 강력한 하드웨어에서 실행되는 컨테이너는 파워가 약한 하드웨어의 인스턴스보다 더 많은 요청을 지원할 수 있습니다.
- 써드 파티 로드 밸런싱 제품과의 통합: 마지막으로 OC4J(전체적으로 Oracle9*i*AS 포함)는 보다 정교한 로드 밸런싱을 위해 Cisco Local Director, BigIP 및 Alteon과 같은 로드 밸런싱 Applicance와 결합할 수 있습니다.
- 연결 재지정: Oracle9*i*AS의 로드 밸런싱 알고리즘을 사용하여 새로운 클라이언트의 요청을 지정하거나 기존의 클라이언트의 요청을 지정하는 인스턴스를 판단할 수 있으며, 동시에 쿠키나 동적 URL 재작성과 같은 표준 기능을 지원하여 클라이언트를 특정 인스턴스의 기존 세션에 바인드하고 재지정합니다. 세션 ID는 쿠키나 동적 URL로 패키지됩니다. 클라언트가 후속 요청을 다시 보내면 로드 밸런서와 HTTP 서버는 요청을 티어를 통해 특정 인스턴스로 재지정할 수 있습니다.

클러스터 지원

Oracle9*i*AS와의 클러스터링은 기본적으로 확장성이 뛰어나고 가용성이 높은 서비스를 투명한 방식으로 제공하기 위해 작업을 조정하는 Oracle9*i*AS 서버 그룹을 사용한다는 것을 의미합니다.

- 무상태(stateless) 클러스터링: 여러 가지 애플리케이션 서버 인스턴스를 클러스터링하여 무상태 요청을 처리하기는 쉽습니다. 요청이 무상태이기 때문에 오라클의 HTTP나 외부 로드 밸런서는 새로운 Oracle9*i*AS 인스턴스에 대한 요청을 동일한 노드나 다른 노드에 지정할 수 있습니다.
- 상태유지(stateful) 클러스터링: 훨씬 어려운 문제는 요청이 상태유지인 경우, 시스템을 클러스터링하는 방법입니다. 예를 들어 e-commerce 서블릿은 HTTPSession Object를 사용하여 메소드 요청 사이의 장바구니 상태를 저장합니다. 클라이언트가 다른 하나의 품목을 장바구니에 추가하면 그 인스턴스에 대한 로드가 너무 높거나 그 인스턴스를 사용할 수 없기 때문에(즉, 실패했기 때문에) 원래 요청을 처리하는 서블릿에 액세스하지 못할 수 있습니다. 이 경우 요청은 HTTPSession Object의 상태가 첫번째 컨테이너에서 복제된 다른 J2EE 컨테이너로 재지정해야 합니다. Oracle9*i*AS의 클러스터링 성능은 애플리케이션 프로그램 코드를 변경할 필요 없이 일한 요구사항을 지원합니다. 이 솔루션이 확장성과 가용성 요구사항을 모두 처리하기 때문에 아래의 가용성 절에 클러스터링 기능을 보다 자세히 설명하기로 합니다.

OC4J와의 클러스터링이란 표현을 Oracle9*i* Database Clustering(Real Application Cluster)과 혼동하면 안됩니다. Oracle9*i*AS가 제공하는 동일한 성능의 정의를 위해 다른 애플리케이션 서버가 사용하고 있기 때문에 이 문서에서는 이 용어를 Oracle9*i*AS의 문맥에서만 사용합니다.

D. OC4J의 고가용성 지원

OC4J에 관련된 고가용성 문제는 다음과 같은 네 가지 영역으로 분류됩니다.

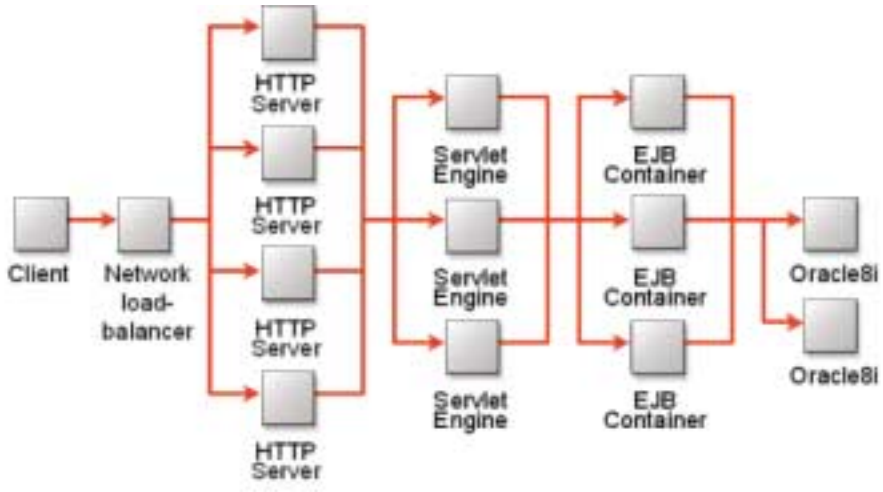
- 단일 장애 지점 없음: 시스템 노드 하나 이상이 고장인 경우에도 Oracle9*i*AS가 기능을 계속 발휘하고 J2EE 클라이언트 요청을 처리하는 구조에서 Oracle9*i*AS 인스턴스를 배치할 수 있는 방법
- 자동 연결 경로 재지정: 특정 인스턴스가 실패할 경우 Oracle9*i*AS가 투명하게 클라이언트를 다른 서버로 경로를 재지정할 수 있는 방법
- 자동 중단 감지 및 재시작: 인스턴스에 대한 평균 복원 시간을 최소화하기 위해 Oracle9*i*AS가 특정 인스턴스의 중단/실패를 감지하여 재시작하는 방법
- 투명 애플리케이션 장애복구와 클러스터 지원: Oracle9*i*AS가 투명하게 J2EE Container의 인스턴스 사이에서 세션 상태를 복제하고 오라클 데이터베이스에서 관리되는 상태의 애플리케이션 Failover를 처리하는 방법

단일 장애 지점 없음

OC4J에는 어떤 단일 장애 지점도 노출하지 않는 아키텍처에서 배치할 수 있는 융통성 있는 배치 모델이 있습니다. 즉, 시스템의 노드 장애도 불구하고 Oracle9*i*AS는 기능을 계속 발휘하고 클라이언트 요청을 처리합니다. 아래의 그림에는 애플리케이션 서버와 데이타베이스를 포함하여 전체 시스템에서 어떤 단일 장애

지점도 없는 Oracle9*i*AS와 Database를 배치하는 예가 표시되어 있습니다. 로드 밸런서는 요청을 복수 오라클 HTTP 서버 중 어느 서버로도 보낼 수 있으며 오라클 HTTP 서버는 어떠한 서블릿 엔진 인스턴스로도 요청을 보낼 수 있습니다. 그러면 Servlet Engine은 요청을 EJB 인스턴스로 보냅니다. EJB 인스턴스는 데이타베이스 연결을 통해 데이타베이스에 액세스합니다. 네트워크 Load Balancer는 고가용성을 위해 중복(이중화) 구성으로 배치되는 Cisco Local Director와 같은 하드웨어 장치입니다.

그림: Oracle9*i*AS 단일 장애 지점 없음



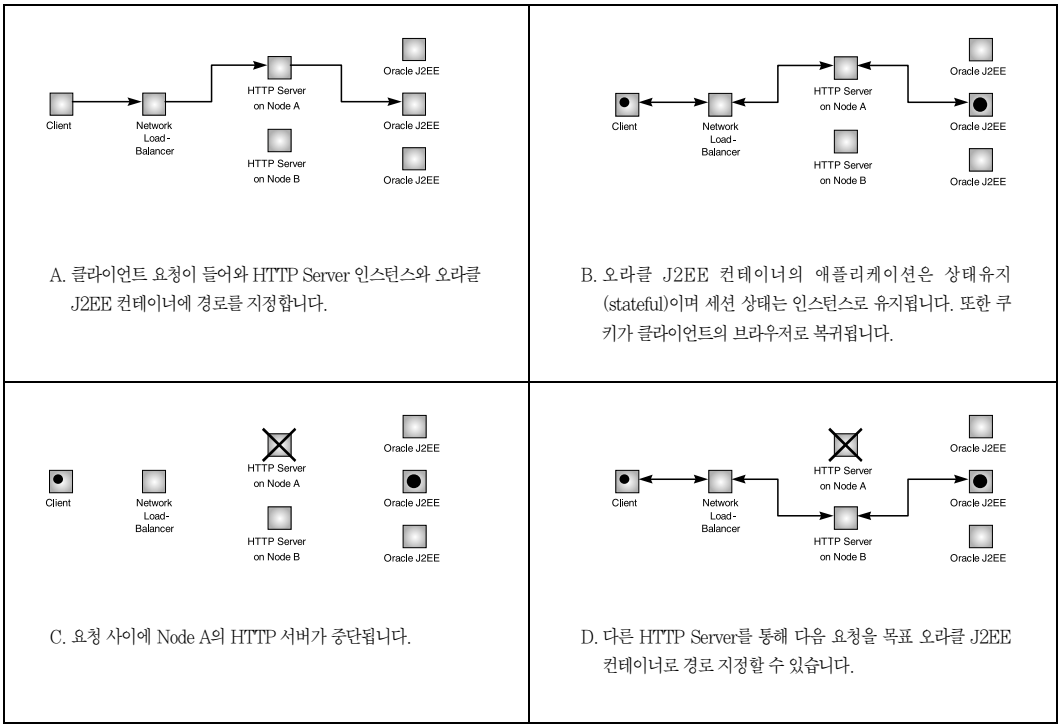
자동 연결 경로 재지정

위의 로드 밸런싱 절에서 설명한 것처럼 Oracle9*i*AS J2EE 배치 중 모든 티어는 적절한 로드 밸런싱 계획을 사용하여 요청을 다른 티어로 경로 지정할 수 있으며 상태유지(stateful) 클라이언트의 연결을 별도의 Oracle9*i*AS 인스턴스에 있는 세션으로 경로를 재지정할 수 있습니다. 요청을 처리 중인 프로세스나 노드가 갑자기 장애를 일으킬 경우 클라이언트가 OC4J의 실행 세션에 액세스를 시도하면, Oracle9*i*AS는 자동으로 투명하게 그 요청을 대체 서버로 경로를 재지정할 수 있습니다.

· 무상태(stateless) 연결 경로 재지정: 완전한 무상태 환경에서 클라이언트는 필수적으로 새로운 클라이언트처럼 보입니다. Oracle9*i*AS의 모든 티어는 다음 티어의 인스턴스가 실패한 사실을 파악하여 로드 밸런싱 알고리즘을 사용, 다음 티어의 다른 하나의 인스턴스를 선택한 다음 연결을 그 인스턴스에 경로 지정합니다. 아래의 그림은 이러한 작업 방법의 한 예입니다.

그림: Oracle9*i*AS 연결 경로 재지정

· 상태유지(stateful) 연결 경로 재지정: 클라이언트가 실패한 Oracle9*i*AS 인스턴스에서 기존의 세션에 재연



결을 시도할 경우 지원되는 세 가지 시나리오는 다음과 같습니다.

- 데이타베이스에 유지되는 세션 상태: 대부분의 J2EE Application Developer는 모든 세션 상태를 복구할 수 있는 지속적 스토어, 오라클 데이타베이스에 보관합니다. 오라클은 미들 티어 Oracle9*i*AS 인스턴스가 완전히 무상태이며, 연결 경로 재지정이 위에서 설명한 방식으로 발생하는 경우 모든 세션을 데이타베이스에 저장할 것을 권장합니다. 애플리케이션은 기본적으로 새로운 JDBC 연결에 특정 커서를 사용하여 데이타베이스에서 세션 상태를 복원합니다(자세한 정보는 아래의 클러스터 항 참조).
- 미드 티어에 유지되는 세션 상태: 일부 애플리케이션 개발자는 세션 상태를 미들 티어에도 유지하지만, 인스턴스가 비정상적으로 종료되는 경우 사용자들이 세션 상태를 상실하기 때문에 안심할 수 없습니다. Oracle9*i*AS는 연결을 특정 J2EE 컨테이너 인스턴스에 경로를 재지정하지만 장애를 감지할 경우, 로드 밸런싱 알고리즘을 사용하여 연결을 다른 인스턴스로 경로 지정합니다(세션 상태에 액세스할 수 없는 경우).
- 클러스터 설정에 복제된 세션 상태: 마지막으로 OC4J가 클러스터링을 지원하도록 구성된 경우 OC4J는 다른 인스턴스의 그룹을 공통 클러스터 ID와 함께 클러스터로 등록합니다. 인스턴스가 실패할 경우 로드 밸런서는 인스턴스 실패를 파악하고 동일한 클러스터 ID가 있는 서버를 확인한 다음 로드 밸런싱 알고리즘을 사용하여 연결을 그러한 서버 중 하나로 경로 지정합니다. 장애 J2EE 컨테이너의 세션 상태는 클러스터의 모든 인스턴스에 복제되므로 클라이언트는 다시 이 세션에 액세스할 수 있습니다.

자동 중단 감지 및 재시작

서버 프로세스가 실패하면 메모리 정리와 실패한 프로세스 재시작 등 Oracle9*i*AS에서 적절한 조치를 취합니다. 결국 Oracle9*i*AS는 Oracle9*i*AS 인스턴스가 실패한 경우 예상치 못한 중단시간을 대폭 줄입니다.

· HTTP 서버: Oracle HTTP Server의 감시 프로세스가 HTTP 자식 프로세스를 모니터링하고 실패한 프로세스를 재시작합니다.

· J2EE 컨테이너: OC4J에서는 J2EE 컨테이너의 실패를 감지하고 해당 인스턴스에 요청을 지정하지 않도록 할 수 있습니다. 컨테이너가 클러스터 구성에 배치된 경우, Oracle9*i*AS에서는 선택한 로드 밸런싱 알고리즘에 따라 연결을 클러스터의 다른 인스턴스에 자동으로 재지정할 수 있습니다. 향후 릴리스에서는 Oracle9*i*AS는 모든 실패한 Oracle9*i*AS 인스턴스를 감지하고 해당 인스턴스를 자동으로 재시작하는 범용 결함 모니터링 서비스를 제공할 예정입니다.

상태 유지(stateful) 및 무상태(stateless) 애플리케이션 장애복구/클러스터 지원

HTTP는 무상태 프로토콜이지만 인터넷에 복잡한 애플리케이션이 배치됨에 따라 개발자들은 대개 이러한 애플리케이션을 구축하고 배치하는 방법에 세션이라는 의미를 추가했습니다. 일반적으로 인터넷 애플리케이션에는 기간, 즉 어떤 상태를 유지해야 하는 시간에 따라 구별되는 다음과 같은 세 종류의 상태가 있습니다.

· 페이지 기간 상태: 기본적으로 단일 요청이 계속되는 동안 존재하는 단기 오브젝트를 가리킵니다. 즉, 보통은 페이지 자체의 쿠키 또는 동적 URL로 클라이언트에 다시 전달됩니다.

· 세션 기간 상태: 단일 페이지보다 오래 지속되지만 세션 기간 동안만 존재하는, 즉 세션 경계에서는 존재하지 않는 세션 상태를 가리킵니다. 전자 상거래의 예에서는 고객이 장바구니에 품목을 넣으면 세션 기간 상태에서 사용자의 세션을 추적하고 장바구니에 넣은 모든 품목을 명시적으로 포착합니다.

· 장기 세션간 상태 지속 상태: 세션 경계에 남아 있는 상태가 지속 상태입니다. 전자 상거래의 예에서는 고객이 주문을 체크아웃하고 전송한 다음 세션을 완료하면 장바구니의 상태가 향후 세션에 대해 지속적으로 보존되어야 합니다. 예를 들어, 고객이 주문 상태를 다시 확인하려고 하면 전송한 특정 주문을 액세스해야 하기 때문에 해당 주문이 보존되어야 합니다.

웹 사이트나 인터넷 애플리케이션의 사용 가능성을 고려할 때, 애플리케이션 서버는 클라이언트(대개 브라우저), 애플리케이션 서버, 데이타베이스 서버로 구성되는 인터넷 하부구조의 일부에 불과하며, 이 하부구조의 모든 구성원들이 협력하여 상태를 유지하고 장애 발생시 효율적으로 복구해야 합니다. 애플리케이션 디자이너는 높은 사용 가능성 문제를 해결하려면 격리된 애플리케이션 서버를 고려하지 않아야 합니다. 높은 사용 가능성이라는 관점에서 인터넷 하부구조에 대한 광범위한 요구사항 세 가지가 있습니다.

· 애플리케이션/사용자 상태 유지: 먼저, 클라이언트, 애플리케이션 서버, 데이타베이스 서버로 구성되는 인터넷 하부구조가 함께 사용자의 상태(페이지 기간 상태, 세션 기간 상태, 지속 상태 중 어느 것인지)를 100% 보존해야 합니다. 인터넷 하부구조의 각 부분에 서로 다른 종류의 상태가 좀더 최적의 상태로 유지되지만 전체 하부구조가 모든 사용자의 상태, 즉 모든 애플리케이션 상태를 유지해야 합니다.

· 애플리케이션 복구 시간 단축: 또한 애플리케이션의 사용 가능성에 가장 중요한 영향을 미치는 요인은 애

플리케이션 실패 이후의 총 복구 시간입니다. 그러므로 하부구조는 1) 모든 시스템 실패를 빨리 감지하고, 2) 사용자를 효율적으로 새 시스템에 지정하거나 복구시키고, 3) 하부구조의 각 부분에서 모든 필요한 애플리케이션 상태를 복구하여 애플리케이션이 실행을 계속하고 사용자들이 자신의 세션을 재개할 수 있도록 하는 세 가지 기능을 제공해야 합니다. 우리는 실패를 감지하고 상태를 복구하고 작업을 재시도하고 시스템을 실패 당시에 실행 중이던 것과 같은 속도로 복원하는 데 걸리는 총 시간을 애플리케이션 복구 시간이라는 용어로 표현합니다.

· 애플리케이션 성능 및 확장성의 성능저하 방지: 끝으로, 높은 사용 가능성을 확보하려면 인터넷 하부구조 때문에 애플리케이션의 전체 성능 또는 확장성이 떨어져서는 안 됩니다. 현재 최고의 여러 애플리케이션 서버조차도 시스템 불안정하게 하고, 애플리케이션 성능을 떨어뜨리고, 효율적인 로드 밸런싱을 방해하는 잘못 디자인된 '클러스터링' 기법을 사용합니다. 잠시 후 오라클의 접근 방식을 이러한 접근 방식과 비교해 봅니다.

상태 관리에 대한 권고

오라클은 이러한 목표를 인식하고 Oracle9*i*AS로 상태 및 높은 사용 가능성을 관리하기 위한 다음과 같은 디자인 지침을 권장합니다.

· 페이지 기간 상태는 클라이언트에 속해야 함: 애플리케이션 서버 대화에 대한 단일 브라우저 지속 시간 동안만 존재하는 상태(즉, 페이지 지속 시간)는 클라이언트에서만 유지되어야 합니다.

· 지속 상태는 데이타베이스에 속해야 함: 세션 경계에서 지속되는 상태는 고가용성이며 복구할 수 있는 지속적 스토어, 즉 데이타베이스에 작성해야 합니다. 데이타베이스는 메모리의 오브젝트 캐싱 및 하드웨어 중단 및 비이중화 디스크 실패의 경우에도 트랜잭션의 보호에 필요한 효율적인 I/O 작업에 아주 유용합니다. J2EE 애플리케이션 인스턴스의 경우 기록의 트랜잭션 엔터프라이즈 빈은 항상 상태를 데이타베이스에 유지해야 합니다. 즉, Oracle9Ias 서버나 애플리케이션에 실패가 발생할 경우에도 애플리케이션의 트랜잭션 상태는 기존의 데이타베이스에서 완전히 복구할 수 있습니다. 또한 OC4J를 사용하면 트랜잭션 방식으로 포착되는 보다 중요한 상태 데이타(예를 들어 30분 세션 동안 구성된 복잡한 웹 주문의 컨텐트)를 포착할 수 있으므로 여러 대의 서버에 장애가 발생하는 경우에도 사용자가 중요한 데이타를 상실할 염려가 없습니다.

· 세션 기간 상태는 데이타베이스에 저장될 수도 있고 메모리에 복제될 수도 있음: 단일 세션 지속 시간 동안만 존재하는 상태는 데이타베이스에 저장하거나 메모리에 복제할 수 있습니다. 각 접근 방식은 성능과 확장성에 어느 정도 영향을 줍니다.

  - 세션 지속 시간 상태를 데이타베이스에 저장: 대부분의 대규모 e-commerce 사이트는 세션 지속 시간 상태를 데이타베이스에 저장합니다. 예를 들어 고객이 장바구니에 품목을 추가하거나 제거할 때마다 Application은 그 상태를 데이타베이스에 입력합니다. 데이타베이스에 이러한 상태를 유지하면 두 가지 중요한 장점을 활용할 수 있습니다. 즉, 1) 첫째, 하나 이상의 애플리케이션 서버에 장애가 발생하는 경우에도 내구성 스토어에서 세션 상태를 복구할 수 있으므로 사용자에게 최고 품질의 서비스를 제

공할 수 있습니다. 2) 둘째, 미들 티어 애플리케이션 서버는 완벽하게 무상태로 유지되므로 애플리케이션 서버를 더 추가하여 미들 티어를 쉽게 확장할 수 있습니다. 즉, 수평 방향 확장성이 제공됩니다. 데이타베이스에 세션 지속 시간 상태를 저장할 때 보편적으로 제기되는 가장 큰 문제들은 다음과 같습니다. 1) 첫째, 세션 상태를 데이타베이스에 입력하면 특히 데이타베이스/디스크 I/O와 같은 데이타베이스 오버헤드가 불필요하게 증가합니다. 사용자는 복구할 수 있는 지속적 스토어에 세션 상태를 백업할 것인지 확인해야 하며 디스크 I/O를 수행해야 합니다. 이때 데이타베이스 확장성 때문에 서비스 품질이 저하됩니다. 2) 둘째, 세션 상태를 데이타베이스에 입력하면 성능 오버헤드가 발생하여 미들 티어와 데이타베이스 사이에 네트워크 속도가 저하되며(대부분의 미들 티어와 데이타베이스 연결성이 100-Base T 네트워크인 경우 무시해도 될 정도임) 연결 구축 오버헤드가 발생하고(일반적으로 연결 풀을 사용하면 제거할 수 있음) 유형 변환이 발생합니다(미들 티어의 자바형 오브젝트에서 데이타베이스의 SQL형으로).

- 중복 메모리 저장(redundant memory storage): 오라클은 위에 설명한 두 가지 문제를 해결해야 하는 필요성을 인식하여 Oracle9Ias로 경쟁력 있는 미들 티어 클러스터링 솔루션을 제공합니다. 이 솔루션이 구성되는 방법에 대한 자세한 내용은 아래에 설명됩니다. 한마디로 고속 IP 멀티캐스트 기능을 통해 자바 오브젝트로 포착한 세션 상태의 공유를 위해 Oracle9Ias 인스턴스 그룹을 구성할 수 있습니다. 오라클은 공유 상태에 대한 공유 디스크 부속 시스템보다는 데이타베이스 I/O 및 오버헤드의 처리, 자바 오브젝트와 같은 상태의 공유(유형 변환 오버헤드 제거) 및 중복 메모리 사용에 집중합니다. 지속 상태를 공유하는데 사용되어서는 안되만, 복수의 동시 서버 기능 장애로부터 보호할 필요가 없는 과도 세션 상태(예를 들어 보안 문맥, 웹 장바구니 컨텐츠)에 효율적으로 사용할 수 있습니다.

Oracle9Ias 클러스터링의 목적

Oracle9*i*AS v.1.0.2.2는 디자인 단계부터 아래에 설명되는 특정 고객 요구사항의 처리를 위해 몇 가지 원칙을 채택했습니다.

· J2EE 애플리케이션 프로그래머에 투명한 고가용성 확보: 몇 가지 애플리케이션 서버는 사용자가 독점적 프로그래밍 API를 사용하여 상태를 공유 메모리 하부구조에 작성하게 함으로써 고가용성 제공을 시도합니다. 오라클은 사용자가 J2EE Application을 특정 애플리케이션 서버 업체의 하부구조에 구속시키길 원하지 않는다는 사실을 잘 알고 있으며, 따라서 애플리케이션에 완벽하게 투명한 클러스터링 기능을 디자인했습니다. 클러스터링 사용을 위해 수행할 필요가 있는 유일한 변경은 애플리케이션 서버 자체에서 배치되는 Application으로부터 완전히 독립적인 클러스터 구성으로 애플리케이션 서버를 구성하는 것입니다.

· OS/하드웨어 플랫폼과 무관하게 고가용성 확보: Oracle9*i*AS v.1.0.2.2를 완전히 자바로 작성한 이후 Oracle9Ias 클러스터는 기본 하드웨어와 운영 체제(OS)에 의존하지 않습니다. 따라서 Oracle9Ias 클러스터는 Microsoft NT나 Linux 또는 대규모이며 더 복잡한 Unix 멀티프로세서를 실행하는 Intel 시스템으로 구성할 수 있습니다. 반대로 다른 애플리케이션 서버는 모든 노드를 동일한 운영 체제에서 실행해야 하며 특정 하드웨어와 OS 플랫폼으로 제한되는 플랫폼 고유 클러스터링 솔루션을 사용합니다. 플랫폼에 의존하지 않는 클러스터링 기능을 제공하기 위해 Oracle9Ias는 IP 멀티캐스트와 같은 신기술에 기초하여

상당한 수준으로 최적화된 프로토콜을 사용합니다.

· 배치 구조와 무관하게 고가용성 확보: 다른 애플리케이션 서버는 클러스터링 지원에 특정 웹 서버나 네트워크 로드 밸런서가 필요합니다. 클러스터링 솔루션에는 웹 서버가 할당되면 안됩니다. 오라클은 Apache를 기본 HTTP 서버로 제공하고 있지만 Oracle9*i*AS 클러스터링 기능은 Microsoft IIS와 Netscape 웹 서버에도 잘 사용할 수 있습니다.

· 구성할 수 있고 확장할 수 있는 클러스터 정의: 또한 경쟁사는 애플리케이션 서버가 세션 장애복구를 위한 클러스터링을 제공한다고 주장하고 있지만 디자인에 심각한 두 가지 한계가 있습니다.

  - *동일하게 구성된 애플리케이션 서버s*: 경쟁사의 일부 애플리케이션 서버는 클러스터링을 지원하기는 하지만 구성할 수는 없습니다. 그 결과 세션 장애복구에 클러스터 솔루션을 사용할 경우 모든 애플리케이션 서버를 동일하게 구성해야 하며 전체 미들 티어가 단일 클러스터를 나타냅니다. 비록 높은 수준의 가용성을 제공하긴 하지만 모든 단일 애플리케이션 서버는 세션 상태를 클러스터 안에 있는 모든 시스템에 복제해야 합니다. 따라서 시스템에서 동시 세션의 수가 증가하면서, 모든 시스템이 다른 모든 애플리케이션 서버 성능, 확장성 및 가용성과의 지속성 유지를 시도함에 따라 서로 다른 시스템 간에 엄청난 통신(chattiness)이 발생하게 됩니다.

  - *확장성을 제한하는 백업 시스템*: 클러스터링을 지원하는 다른 애플리케이션 서버들은 이러한 한계를 해결하기 위해 모든 시스템이 하나의 시스템(즉, 버디)를 사용하여 상태를 이 시스템에 복제하는 버디 시스템으로 이동했습니다. 버디 시스템을 사용하면 시스템 간의 통신(chattiness)은 줄일 수 있지만 다음과 같은 중요한 두 가지 한계가 있습니다. 즉, 1) 가용성이 훼손됩니다. 양쪽 시스템에 동시 장애가 발생할 경우 양쪽 시스템의 모든 세션 상태가 손실됩니다. 2) 둘째, 두 시스템을 모두 버디로 구성한 경우에도 그 중 하나는 항상 일차 상태 서버로 구성해야 합니다. 클러스터의 모든 서버는 모든 사용자 요청이 도달하는 일차 상태 서버에 대한 액세스에 병목 현상을 일으킵니다. 이 구조로는 로드 밸런싱 기술을 사용할 수 없으므로 확장성이 손상됩니다.

  - *정적 IP 기반 멀티 캐스트 기능*: 마지막으로 클러스터링을 사용하는 경쟁사의 애플리케이션 서버는 상태를 발표하고 수신하기 위해 상태 복제 사용 IP 주소에 대해 IP 멀티캐스트를 사용합니다. 정적 IP 주소에 대한 장애복구를 바인딩함으로써 어떤 로드 밸런싱 기술도 사용할 수 없으며, 그에 따라 확장성이 손상됩니다.

Oracle9*i*AS 클러스터링 기능은 이러한 디자인 한계를 동시에 모두 해결합니다. Oracle9*i*AS는 클러스터링을 위해 미들 티어 애플리케이션 서버를 다른 클러스터로 구성합니다. 서버 클러스터의 모든 서버는 동질 서버이지만 모든 미들 티어 시스템은 동일한 클러스터에 굳이 속할 필요는 없습니다. 클러스터링을 구성할 수 있는 것으로 작성함으로써 Oracle9*i*AS는 동일하게 구성된 모든 시스템에 공통인 통신(chattiness) 문제를 해결할 수 있는 동시에 클러스터의 모든 시스템에서 로그 밸런싱을 계속 운영할 수 있습니다. 또한 Oracle9*i*AS는 정적 IP 주소가 아닌 동적 주소를 사용하여 로드 밸런싱과 확장성 기술을 사용할 수 있는 동시에 높은 가용성을 제공합니다. 마지막으로 시스템 관리자는 Oracle9Ias의 자동 경로 지정 기능을 활용하여 필요할 경우 클러스터 안에서 비즈니스 처리를 동적으로 다시 분할할 수 있습니다.

· 클러스터링과 투명 애플리케이션 장애복구의 결합: 마지막으로 다른 모든 애플리케이션 서버는 미들 티어에 유지되는 상태에 대해 상태유지(stateful) 장애복구만 취급하지만 Oracle9*i*AS은 오라클 데이타베이스 안에서 관리되는 지속 상태에 대한 장애복구 기능으로 미들 티어에서 관리되는 세션 상태를 위한 Transparent Application Failover를 제공하므로 엔드 투 엔드 가용성이 100% 보장됩니다.

· 사용하기 쉬운 클러스터링과 고가용성을 낮은 원가로 제공: 다른 모든 애플리케이션 서버는 1) 클러스터링 목적을 위해 고가의 특수 하드웨어를 구매하고, 2) 클러스터링 목적을 위해 고가의 특수 제품 버전을 구매하며 3) 클러스터링 구성을 위해 매우 복잡한 구성 설정과 구조를 사용해야 합니다. 반대로 Oracle9*i*AS의 구성 기능에는 Oracle9*i*AS Standard Edition의 일부로서 구매 즉시 사용할 수 있는 기능을 제공합니다.Oracle9*i*AS에는 특수 하드웨어의 사용이 필요하지 않으며 수행하기 쉬운 다섯 단계로 구성 파일의 세 가지 파라미터를 설정함으로써 구성할 수 있습니다.

Oracle9*i*AS 클러스터링 기능

지금까지 Oracle9*i*AS 세션 장애복구 기능의 구성에 사용되는 디자인을 설명했습니다. 이제 구성을 자세히 살펴보도록 합니다.

· 웹 프리젠테이션 및 웹 서버 장애복구: Oracle9*i*AS 클러스터는 Apache, Netscape 또는 Microsoft IIS Server와 같은 표준 웹 서버 뒤에 위치할 수 있습니다. 웹 브라우저의 HTTP 요청은 웹 서버가 Oracle9*i*AS Servlet/JSP 엔진으로 전달합니다. 클러스터의 처음 줄은 웹 클라이언트와 웹 서버 사이의 DNS Round Robin을 사용합니다. 인터넷의 Domain Name Service인 DNS는 사이트의 웹 서버를 위한 IP 주소 목록에 웹 사이트의 이름을 나열합니다. DNS는 조회 요청을 받을 때마다 주소 목록을 검색합니다. 일반적으로 웹 클라이언트는 DNS가 제공하는 목록의 처음 웹 서버와 연락합니다. 일정한 시간이 경과한 후 서버가 실패하는 경우 클라이언트는 DNS 요청을 다시 하여 새로운 서버를 사용합니다. 따라서 로드 밸런싱 및 장애복구가 단순하게 이루어집니다. 예를 들어 웹 서버 로드를 고려하고 DNS가 리턴한 목록에서 실패한 서버를 제거해주는 보다 정교한 IP 수준 로드 밸런싱 및 장애복구 스키마를 설치할 수도 있으며, 특정 클라이언트의 요구가 동일한 웹/애플리케이션 서버(모듈 실패)로 항상 재지정되도록 할 수 있습니다. Oracle9*i*AS는 Cisco Local Director 및 BigIP와 같은 표준 로드 밸런싱 장치에 사용하도록 인증되었습니다.

· 세션 상태 복제 및 클러스터 장애복구: 클러스터링의 두 번째 줄은 애플리케이션 서버에서 동적으로 생성된 페이지를 위한 것입니다(예를 들어 서블릿 엔진). 다음과 같은 단계를 따라 진행합니다.

  - Oracle9*i*AS 서버의 클러스터 구성: Oracle9*i*AS 애플리케이션 서버 세트는 클러스터 아일랜드로 등록됩니다. 즉, 모든 애플리케이션 서버를 동일하게 구성할 필요는 없지만 3-4 애플리케이션 서버를 동질 시스템으로 구성할 수 있으며 서로에 대해 장애복구 기능을 제공할 수 있습니다. 전체 미들 티어는 필요한 경우 다른 클러스터 아일랜드 세트로 구성할 수 있습니다. 애플리케이션 서버를 클러스터에 속하도록 구성하는 일은 세 가지 클러스터 파라미터로 XML 구성 파일을 편집해야 하는 Oracle9*i*AS v.1.0.2.2에게는 상당히 단순한 작업입니다.
  - 로드 밸런싱 및 경로 지정: 클라이언트가 처음 로드 밸런서와 접촉하면 요청이 클러스터 안의 애플리케이션 서버로 향합니다. 그 애플리케이션 서버에서 세션이 작성됩니다. 이 세션은 구성할 수 있는 시

간 초과 설정이 만료될 때까지 지속되며 로드 밸런서는 세션 지속 시간 중 클라이언트에서 동일한 애플리케이션 서버 인스턴스로 후속 요청의 경로가 다시 지정됨을 압니다. 일차/주 애플리케이션 서버에 장애가 발생하면 로드 밸런서가 그 인스턴스가 실패했음을 파악하여 투명하게 클라이언트를 클러스터 안의 다른 애플리케이션 서버로 경로 재지정합니다. 이 경우 표준 로드 밸런싱 알고리즘에 계속 사용되어 요청을 클러스터 안의 여러 노드로 분배합니다.

- 복제된 세션 상태: HTTPSession Data와 ServletContext Data, 이 두 가지 세션 상태가 IP-Multicast를 통해 첫째 애플리케이션 서버에서 클러스터의 다른 서버로 복제됩니다. 고객 장바구니의 상태가 첫번째 애플리케이션 서버의 서블릿에서 HTTPSession 오브젝트로 유지되는 상황을 예로 들어보겠습니다. 첫 번째 애플리케이션 서버에서 장애가 발생하면 클라이언트의 요구는 투명하게 클러스터 안의 다른 애플리케이션 서버로 경로 재지정되며 장바구니 상태는 하나의 애플리케이션 서버에서 메모리 내부 복제를 사용하는 다른 애플리케이션 서버로 장애 복구됩니다. 복제 시스템은 효율적이며 갱신된 차이만 일차 서버에서 클러스터의 다른 서버로 전달합니다.

- Transparent Application Failover(TAF) 및 데이타베이스 상태 관리: 마지막으로 모든 Oracle9*i*AS 미들 티어는 작은 수의 연결에 있는 복수 클라이언트를 데이타베이스로 다중화하여 성능과 확장성을 향상시키는 연결 풀을 사용하여 Oracle Database Server와 통신합니다. 모든 애플리케이션은 오라클 데이터베이스에 지속적으로 장기간 존속하는 상태를 작성합니다. Oracle Database가 클러스터 구성으로 배치되고 데이타베이스의 특정 노드가 실패할 경우 데이타베이스에서 유지되고 있는 모든 상태는 클러스터의 다른 노드로 투명하게 장애 복구합니다. Oracle9*i*AS를 Oracle Database와 함께 사용하면, 미들 티어 서버가 JDBC 연결과 데이타베이스 요청을 장애 복구된 노드로 투명하게 경로 재지정할 수 있습니다. 연결이 설정된 방식에 따라 Oracle9*i*AS는 데이타베이스에 다양한 정도의 투명 애플리케이션 장애복구(TAF)를 제공할 수 있는데, 지속 상태의 cold, warm, hot 장애복구가 있습니다.

5. 결론

요약하여 설명하면 Oracle9*i*AS v.1.0.2.2는 오라클의 자바 전략에 있어 상당한 진전을 의미합니다. 이 릴리스는 완전히 자바로 작성되어 표준 Java Development Kit(JDK) Virtual Machine(Java VM)에서 실행되는 완벽한 자바2 Enterprise Edition(J2EE) 컨테이너인 OC4J를 제공합니다. OC4J 주요 기능은 다음과 같습니다.

· JDK에서 실행하는 순수 자바 컨테이너/런타임: 먼저 OC4J는 완전히 자바로 구현됨에 따라 다음과 같은 장점을 발휘합니다. 1) 경량화 15Mb 디스크, 20Mb 메모리, 2) 신속한 설치 5분 이내, 3) 간편한 사용 표준 자바 개발 및 프로파일링 툴 지원, 4) Solaris, HP-UX, AIX, Tru64, Windows NT 및 Linux와 같은 모든 표준 운영 체제 및 하드웨어 플랫폼에서 사용가능. Oracle9*i*AS v1.0.2.2는 JDK 1.2.2와 JDK 1.3 자바 VM에서 실행되도록 검증되었습니다.

· 완벽한 J2EE 1.2 컨테이너: 둘째, JSP Translator, 자바 서블릿 엔진 및 Enterprise JavaBeans(EJB) 컨테이너와 같은 완전한 J2EE 컨테이너을 제공합니다. 또한 자바 Messaging Service, JMS 및 기타 몇 가지

자바 사양을 지원합니다. OC4J는 Enterprise JavaBeans(EJB) 1.1, Servlet 2.2, 자바 Server Pages(JSP) 1.1, JTA 1.0.1, JNDI 1.2, JMS 1.0, JDBC 2.0 및 자바Mail 1.1.2와 같은 모든 J2EE 1.2 API에 대해 완벽한 지원을 제공합니다. 또한 OC4J는 J2EE 1.3 사양의 일부인 EJB 2.0과 Servlet 2.3 사양의 구현을 제공합니다. 또한 JSP Tag 라이브러리, WAR 및 EAR 파일 기반 배치와 같은 표준 서비스와 J2EE Application의 자동 배치 및 신속한 배치를 지원합니다.

· 경쟁력 있는 Footprint, 성능/확장성/가용성: 마지막으로, OC4J는 Oracle9*i*AS v1.0.2.2는 개발자에게 다음과 같은 여러 가지 이점을 제공합니다.

- Footprint: OC4J의 디스크와 메모리 footprint는 작으므로 설치, 구성 및 사용이 매우 쉽습니다.
- 빠른 성능: OC4J는 업계에서 가장 빠른 자바 애플리케이션 서버입니다. 별도의 문서에서 업계의 주요 경쟁사와의 벤치마크 결과를 제공할 것입니다.
- 확장성과 고가용성: 훨씬 적은 메모리 용량으로도 가능합니다. 단일 장애 지점이 없으며 상태유지 및 무상태 애플리케이션 장애복구를 모두 처리하기 위해 정교한 IP-Multicast 기반 클러스터링 기능을 제공하며 Transparent Application Failover와 결합하여 업계 최고의 가용성을 제공합니다..

Oracle9*i* 애플리케이션 서버

자바2 Enterprise Edition 기능 및 디자인에 대해 고려할 사항

2001년 5월

ORACLE®

한국오라클(주)
서울특별시 강남구 삼성동 144-17
삼화빌딩
대표전화 : 2194-8000
FAX : 2194-8001

한국오라클교육센타
서울특별시 영등포구 여의도동 28-1
전경련회관 5층, 7층
대표전화 : 3779-4242 4
FAX : 3779-4100 1

대전사무소
대전광역시 서구 둔산동 929번지
대전둔산사학연금회관 18층
대표전화 : (042)483-4131 2
FAX : (042)483-4133

대구사무소
대구광역시 동구 신천동 111번지
영남타워빌딩 9층
대표전화 : (053)741-4513 4
FAX : (053)741-4515

부산사무소
부산광역시 동구 초량동 1211 7
정암빌딩 8층
대표전화 : (051)465-9996
FAX : (051)465-9958

울산사무소
울산광역시 남구 달동 1319-15번지
정우빌딩 3층
대표전화 : (052)267-4262
FAX : (052)267-4267

광주사무소
광주광역시 서구 양동 60-37
금호생명빌딩 8층
대표전화 : (062)350-0131
FAX : (062)350-0130

고객에게 완전하고 효과적인
정보관리 솔루션을 제공하기 위하여
오라클사는 전 세계 145개국에서
제품, 기술지원, 교육 및
컨설팅 서비스를
제공하고 있습니다.

http://www.oracle.com
http://www.oracle.com/kr