Enterprise JavaBeans(EJB) 컴포넌트의 성능 측정 방법

오창남[0] 이긍해
한국항공대학교 컴퓨터공학과
{freedom, khlee}@mail.hangkong.ac.kr

Performance Measuring Method of Enterprise JavaBeans(EJB)

Chang-Nam Oh[O] Keung-Hae Lee
Dept. of Computer Engineering, Hankuk Aviation University

요 약

IBM사의 San Francisco 프로젝트 성공이후 컴포넌트 시장은 성장하고 있다. 컴포넌트 기반 개발 방법의 사용이 보편화되면 응용 프로그램은 필요한 컴포넌트들을 조립하여 개발하게 될 것이다. 컴포넌트는 응용프로그램의 성능에 많은 영향을 미치게된다. 기존의 방법만으로는 컴포넌트의 특징이 고려되지 않기 때문에 분산 컴포넌트의 성능을 비교하는 데에 적합하지 않다. 기존의 컴포넌트 측정방법에는 컴포넌트의 응답시간, 효율(throughput), 메소드(method) 처리 시간, 풀(pool)의 개수에 따른 에러 발생률등이 있다. 본 논문에서는 분산응용을 위한 컴포넌트의 성능을 측정하는 추가적인 방법을 제안한다. 첫째 각 빈들의 처리 응답시간이다. 둘째 트랜잭션의 응답시간이다. 셋째 풀(pool)안에서 객체 개수에 따른 응답시간이다. 객체 수에 따른 컴포넌트 처리 시간이다. 넷째 컴포넌트 알고리즘 처리시간이다. 다섯째 힙(heap) 사용률이다. 이에 컴포넌트 구매자는 성능에 대한 객관적인 데이터로 보고 선택적으로 컴포넌트를 구입할 수 있다.

1. 서론

객체지향 개발 방법만으로는 점차로 보편화되는 대규모 분산환경의 소프트웨어를 효과적으로 개발하기에 한계가 있다. 1994년부터 1996년까지 추진된 IBM SanFrancisco 프로젝트 프로토타입은 컴포넌트 기반 개발 방법을 적용해서 인터넷 기반의 이비즈니스(E-business) 솔루션을 구축한 프레임웍이다. 객체지향 기술 및 자바 기술을 기반으로 1200여개의 비즈니스 컴포넌트를 제공했다. 이것은 컴포넌트 개발 방법론이 객체지향 방법의 취약점을 개선한 좋은 적용사례다.

OVUM 사에 의하면, 컴포넌트 시장은 1998년 30억$지만 2002년에는 640억$로 성장할 것으로 전망한다. 또 Gartner 사는 컴포넌트 시장이 2001년 최소 60% 성장하며, 향후 년간 30%이상 증가할 것으로 예측하고 있다. 컴포넌트 시장이 성장하게되면, 응용 프로그램은 필요한 컴포넌트들을 조립하여 개발하게 될 것이다[1].

컴포넌트를 사용하여 개발된 응용프로그램은 컴포넌트의 성능에 따라 영향을 받는다. 응용프로그램 개발자는 컴포넌트를 선택하기 전에 그 컴포넌트의 성능을 알아야 한다. 이때 컴포넌트 성능에 대한 자료는 객관성을 유지해야 한다[24].
컴포넌트 기반 소프트웨어 개발을 위한 테스팅 방법은 기존의 방법과 다르다. 이에 컴포넌트를 테스트하는 연구가 여러 연구실에서 진행중이다[3].

컴포넌트 성능측정방법도 컴포넌트 기반 소프트웨어를 위한 테스트 방법이 다른 것처럼 성능측정방법을 적용해야한다. 기존 성능측정방식은 바이너리 형태의 독립된 모듈단위인 컴포넌트 단위로 성능측정하기 보다는 응용프로그램 레벨에서 응용프로그램의 기능 및 제어들을 테스트하였다. 컴포넌트는 특정 컴포넌트 아키텍처를 기반으로 된 것이기 때문에 아키텍처에 따라 측정방법이 다르게 된다. 분산 응용 프로그램에 적합한 컴포넌트를 측정하기에 분산환경을 고려하지 않은 기존 성능 측정 방식에 추가적인 측정방법이 있어야한다[3][10][11].

현재 Sun 사의 EJB는 전세계적으로 분산 응용을 위한 컴포넌트의 표준 표준으로 자리잡아가고 있으며, 국내에서도 EJB 기반으로 정보시스템을 구축하고 있는 사례들이 늘어나고 있다[4]. 본 논문에서는 EJB 아키텍처 기반 컴포넌트의 성능 측정에 관해서 설명한다.

현재 컴포넌트 측정방법으로 컴포넌트의 응답시간, 효율성(throughput), 메소드(method) 처리시간, 풀(pool)의 개수에 따른 예외 발생률등이다.

본 논문에서는 분산환경에서 컴포넌트 성능을 측정하는 추가적인 방법론을 제안한다. 첫째 각 빈들의 처리 응답시간이다. 둘째 트랜잭션의 응답시간이다. 셋째 풀(pool)안에서 객체 개수에 따른 응답시간이다. 넷째 알고리즘 처리시간이다. 다섯째 힙(heap) 사용률이다. 2장에서는 컴포넌트에 대한 개념 및 특징과 EJB을 설명한다. 3장에서는 기존의 성능 측정 방법에 대해서 설명한다. 4장에서는 본 논문에서 제안하는 컴포넌트의 성능측정방법을 제시한다. 5장에서는 결론 및 향후 연구에 관해

서 논한다.

2. 컴포넌트의 정의와 특징 그리고 EJB 아키텍처

객체지향 기술을 대규모 시스템 개발에 적용하기 위해 출현한 컴포넌트 기술은 앞으로 소프트웨어 기술을 혁신적으로 변화시킬 것으로 기대되고 있다. 여기서는 먼저, 컴포넌트 분야에 있어서 본 연구와 관련된 기존 연구를 살펴본다.

2.1 컴포넌트의 정의와 특징

컴포넌트는 개발 프로세스에 있어서 재사용 가능하고 분리 가능한 제품이다. 그러나, 컴포넌트에 대한 정의는 다양하다. Gartner 그룹은 '컴포넌트는 동적으로 바인딩 할 수 있는 하나 이상의 프로그램을 하나의 단위로 패키지한 것으로 실행시간에 인터페이스를 통해서 접근한 것'이라고 정의한다. Kozaczynski는 '컴포넌트는 자발적인 비즈니스 객체 또는 비즈니스 로직을 소프트웨어로 구현한 것이다.' 라고 말한다[2].

컴포넌트에는 다음과 같은 특징이 있다. 첫째, 컴포넌트는 모듈화되어 있는 단위이다. 둘째, 컴포넌트는 개발시 직접적으로 사용될 수 있도록 인터페이스를 가져야 한다. 셋째, 컴포넌트는 그 내용을 확인할 수 있는 의미가 있게 뚜렷이 정의된 단위이다. 넷째 컴포넌트는 아키텍쳐를 기반으로 한다. 다섯째, 컴포넌트는 커스터마이징(customizing)이 가능해야 한다. 여섯째, 컴포넌트는 그의 원래 내용으로부터 분리될 수 있어야 한다[2].

2.2 EJB 컴포넌트 구조

EJB 구조는 보통 클라이언트, EJB 서버, 데이터베이스의 3-tier로 구성되어 있다. EJB 컴포넌트는 미들티어(middle-tier)인 EJB 컨테이너 안에 있는 빈으로 존재한다. EJB 컨테이너는 EJB 서버 안에 위치하게 된다. EJB 모델은 분산 객체 관리, 트랜젝션, 보안, 영속성등을 지원한다[6].

모든 EJB 빈은 빈의 생성에 관련된 기능을 가지고 있는 홈(Home) 인터페이스와 비즈니스 로직을 가지고 있는 원격(Remote) 인터페이스를 가지며, 클래스가 지원하는 기능과 상태, 자료의 특징에 따라 세션 빈(Session Bean)과 엔티티 빈(Entity Bean)으로 나뉘게 된다[5].

2.2.1세션 빈과 엔티티 빈

EJB에서는 빈을 기본적인 컴포넌트 단위로 정의하고 있다. 클래스들이 가지고 있는 데이터의 지속성이 일시적일 때 세션 빈을 사용하고, 영구적일 때는 엔티티 빈으로 구현한다.

세션 빈(Session Bean)

하나의 클라이언트를 위해 실행되며, 트랜잭션에 참여할 수 있다. 상대적으로 주기가 짧아서 EJB 컨테이너가 기능을 멈추면 제거된다. 세션 빈은 다시 상태유지 세션 빈(Stateful Session Bean) 과 무상태 세션 빈 (Stateless Session Bean)으로 나눠진다.

상태유지 세션빈: 상태유지 세션빈은 클라이언트 대신 작업을 수행하고, 그 클라이언트와 관련된 상태를 유지한다. 상태유지 세션빈에 의해 호출된 메소드들은 클라이언트와 세션빈 사이의 대화상태(Conversational State)로부터 데이터를 읽고 쓸 수 있다.

무상태 세션빈 : 무상태 세션빈은 메소드로 표현되는 서비스들의 집합이다. 빈은 한번의 메소드 호출이 일어난 후 다음번 호출까지 상태를 유지하지 않는다. 어떤 요청이 있을지를 상관하지 않고, 호출된 메소드를 실행한 다음 결과를 반환한다.

엔티티 빈(Entity Bean)

엔티티 빈의 경우 자료의 지속성을 구현하는 방법에 따라 컨테이너 관리 빈(Container managed Bean)과 빈 관리 빈(Bean managed Bean)이 있다.

컨테이너 관리 빈 : 컨테이너 관리 빈은 EJB 컨테이너에서 기본키(Primary Key) 클래스를 이용하여 지속성을 자동적으로 관리한다. 따라서 빈 개발자가 별도로 지속성을 유지할 필요는 없다.

빈 관리 빈 : 빈 관리 빈은 컨테이너 관리 빈과 달리 빈 개발자가 직접 데이터베이스에 접근하여 데이터의 지속성을 구현여야 한다.

2.2.2 배치 서술자(Deployment Descriptor 또는 DD)

배치 서술자는 EJB-jar 파일 생산자와 소비자 사이 간 계약의 일부로 구조적 정보, 조립정보등이 포함된다. 빈 제공자에 의해 만들어진 하나의 EJB-jar 파일은 하나 이상의 엔터프라이즈 빈들을 포함하며, 보통 응용프로그램 조립에 관련된 지시사항들은 포함하지 않는다.

3장 성능 측정 방법

현재 EJB 아키텍처 기반으로 개발된 컴포넌트의 성능을 측정하는 방법에 관한 연구는 아직 많지 않다. 다음에는 본 논문과 관련된 기존의 연구를 요약한다.

3.1 컴포넌트 성능 측정 방법

첫째, 효율성(throughput)이다. 두 번째 빈의 평균 처리응답시간에 대한 역이다. 즉, 접속한 클라이언트 수에 따른 컴포넌트가 초당 처리하는 시간이다. 단위시간당 얼마나 빨리 클라이언트의 요구사항을 만족시키는지를 측정한다.

둘째, 처리응답시간(Response time)이다. 이 측정방법은 빈 하나를 생성하고, 그 빈을 수행하는데 걸리는 총 시간이다. 그리고 클라이언트 수 증감에 따라 컴포넌트에서 처리하는데 걸리는 시간을 측정한다.

셋째, 접속한 클라이언트 수에 따른 예외 발생률(Exception)이다. 클라이언트의 수가 많을수록 컴포넌트에서 처리해야 할 부분은 증가하게되므로 예외발생률도 높아질 것이다. 사용자

수가 증가함에 따라서 예외 발생률이 증가하는 것을 나타낸다.

넷째, 메소드(method)에 따른 초당 처리 시간(Response time/ method)이다. 컴포넌트 안에는 각 기능에 따라서 여러 가지 메소드들이 있다. 해당 메소드들을 수행하는데 소요되는 시간을 측정한다. 메소드들의 처리 시간을 측정한다[7][8].

3.2 자바 프로그램 성능 측정 방법

자바 프로그램의 성능 측정 방법은 그 자체로 EJB 성능 측정을 위한 것은 아니다. 그러나 EJB가 Java 기반의 아키텍처를 가지고 있기 때문에 Java 프로그램의 성능측정 방법을 이용할 수 있다. 이 분야의 기존 연구는 자원의 사용률에 관한 정보를 가지고 자바 프로그램의 성능측정을 한다. CPU 활용도, Memory 사용률, 메모리 낭비(memory leak), 성능 병목현상을 발견한다[9].

4. 추가적인 컴포넌트 성능 측정 방법

컴포넌트 성능을 측정하는데 필요한 측정방법을 제안한다.

첫째, 각 빈들의 처리 응답시간이다. 기능을 수행할 때, 빈(bean) 하나만으로 가능한 것은 아니다. 보통 두개이상의 빈들을 조합해서 처리를 한다. 현재 처리시간(Response time) 측정 방법은 클라이언트가 처리요청을 하고, 그 요청한 것이 완전히 처리되는 시간을 측정하는 방법이다. 이 방법은 컴포넌트를 처리할 때, 어느 빈에서 로드가 많이 걸리는지 측정할 수 없는 문제점이 있다. 빈들사이의 상호작용시, 특정 빈에서 처리시간이 많이 소요된다면 그 빈에서 병목현상이 일어난다는 것이다. 이런 병목현상이 발생하는 빈을 발견하기위해 각 빈들의 처리시간을 알아야 한다.

둘째, 트랜잭션의 응답시간이다. 엔티티 빈은 데이터의 지속성을 유지하기위해서 데이터를 데이터베이스에 저장하거나 읽어온다. 데이터베이스에서 데이터를 처리하는 시간은 비즈니스 로직부분에서 많이 소요되는 부분이다. 데이터베이스를 잘 설계되어있는지를 알기위해서는 처리시간을 확인해야 한다. 제안하는 방법은 실재 데이터베이스를 통한 트랜잭션 처리 시간을 측정하는 것이다. 트랜잭션 처리 시간에 따른 컴포넌트 성능을 측정한다.

셋째, 알고리즘 처리시간이다. 빈의 종류에 따라 빈이 생성되는 시점이 다르다. 무상태 세션빈과 엔티티 빈은 서버가 운용되면서 미리 풀(Pool)갯수만큼 빈의 인스턴스를 생성한다. 상태유지 세션빈은 사용자로부터 빈 생성 명령이 요구되면 그때부터 빈 생성 작업을 시작한다. 이런 경우를 현재 처리응답시간 측정방법은 동일한 기능을 하는 두 컴포넌트의 처리응답시간으로 컴포넌트들간의 성능비교를 할 수 없다. 따라서 서로 다른 빈들의 처리시간을 비교하기위해서는 동일한 조건을 두어야 한다. 따라서 컴포넌트 알고리즘 처리 응답 시간을 구한다. 동일한 조건에서 빈의 알고리즘 처리 시간을 비교해보면 더 나은 알고리즘 성능을 알 수 있다.

넷째, 풀(pool) 크기에 따른 처리시간 및 자원활용이다. 빈 생성시, 풀에 미리 빈의 인스턴스를 생성할 수 있다. 생성된 빈은 클라이언트가 요구를 했을 때 인스턴스를 생성하는 것이 아니라, 미리 생성된 인스턴스를 사용할 수 있도록 한다. 따라서 클라이언트가 요구했을 때 빈을 생성하는 시간이 소모되지 않는다. 따라서 많은 클라이언트가 컴포넌트 처리를 요구해 왔을 때, 풀 크기가 클수록 빈을 생성해서 처리하는 것보다 빠르다. 하지만 미리 생성된 빈 인스턴스는 시스템 자원을 그만큼 사용한다. 풀 크기는 처리시간과 시스템 자원간 상관관계를 가진다. 따라서 풀 크기에 따른 컴포넌트 처리시간과 시스템 자원 활용을 측정한다.

다섯째, 힙(heap) 사용률이다. 해당 컴포넌트을 처리하기위해 시스템은 힙 크기를 할당한다. 기본적으로 해당 컴포넌트를 활용하기 위해서는 최소 힙 용량이 있어야한다. 빈 사용을 위한 할당된 전체 힙 크기를 측정한다. 또한 현재 빈에서 사용하고 있는 힙과 사용가능한 힙 크기를 측정한다.

5. 결론 및 향후 연구

컴포넌트의 사용이 증가함에 따라 컴포넌트의 개발자와 사용자는 다른 역할을 하게된다. 사용자들이 컴포넌트를 구입하기 위해서는 각 컴포넌트의 성능에 대해 비교하기 위한 방법이 요구된다.

본 논문은 EJB 컴포넌트의 성능 측정 방법을 제안했다. 향후 연구로는 제안한 컴포넌트 성능측정 방법을 지원하는 성능측정 도구를 설계 구현하겠다. 컴포넌트의 객관적인 성능을 비교하는 것이 가능하게 되면 사용자는 동일 기능을 가진 다수 컴포넌트 중에서 우수한 컴포넌트를 선택할 수 있을 것이다.

6. 참고 문헌

[1] "Worldwide total packaged software", IDC, 1999

[2] 김수동, "컴포넌트정의 및 관련기술동향", 정보처리학회 소프트웨어공학회지,12권 3호, pp.3-18, 9.1999

[3] Jerry Ga, Eugene Y. Zhu, Simon shim, "Testing component-based software ", STARWEST '99

[4] 권오천, "공용컴포넌트 플랫폼 선정시 고려사항", 공용컴포넌트플랫폼선정을위한 공청회,pp.71-73, 3.2000

[5] 김수동, "Enterprise JavaBeans(EJB) 기반의 컴포넌트프로그래밍",정보처리학회지,7권 4호 ,pp.40-45, 7.2000

[6] Ed Roman, "Mastering Enterprise JavaBeans and the Java2 Platform", Enterprise Edition, pp 263-302, WILEY, 1999

[7] "EJB-test.com.", URL http://www.testmybeans.com/, TestMyBeans Inc.

[8] "QA lab",URL http://www.flashline.com/, Flashline Inc.

[9] "OptimizeIt 3.1", URL http://www.optimizeit.com/, Intuitive Systems Inc.

[10] Elaine J. Weyuker, "Testing Component Based Software : A Cautionary Table", IEEE, pp.54-59, 10.1998

[11] Roy S. Freedman, "Testability of Software Components", IEEE, pp.553-564, 6.1991