2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

Enterprise JavaBeans(EJB) 컴포넌트의 성능 측정 시스템 설계

오창남, 이긍해
한국항공대학교 컴퓨터공학과
e-mail:{freedom, khlee}@mail.hangkong.ac.kr

A Design of Performance Measuring System for Enterprise JavaBeans(EJB)

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

요약

대규모 분산환경 소프트웨어 개발을 위한 컴포넌트의 사용이 점차로 증가하고 있다. 응용프로그램 개발시 사용되는 컴포넌트는 응용프로그램의 성능에 큰 영향을 미친다. 컴포넌트 시장이 성숙되면 선택 가능한 다수 컴포넌트를 비교 선택할 수 있는 방법이 필요하게 된다. 컴포넌트의 성능 측정에 기존 성능 측정 방식을 사용하기 위해서는 추가적인 방법이 필요하다. 본 논문에서는 분산응용을 위한 컴포넌트의 성능 측정을 하는 요소를 제안한다. 빈들의 처리 응답시간, 트랜잭션의 응답시간, 컴포넌트 알고리즘 처리시간, 힙(heap) 사용률, 풀 크기에 따른 CPU사용률을 컴포넌트 성능 비교를 위한 방법으로 제안하며 그 측정을 위한 시스템을 설계한다.

1. 서론

1994년부터 1996년까지 추진된 IBM SanFrancisco 프로젝트는 객체지향 기술 및 자바 기술을 기반으로 하여 1200여개의 비즈니스 컴포넌트를 제공했다. 이것은 컴포넌트 개발 방법론이 객체지향 방법의 취약점을 개선할 수 있음을 보여 주는 좋은 적용사례다[2].

Gartner 는 컴포넌트 시장이 2001년 최소 60% 성장하며, 향후 년간 30%이상 증가할 것으로 예측하고 있다[1]. 컴포넌트 시장이 성장하게되면 응용프로그램 개발회사는 프로그램의 개발 일정과 비용을 감소시키기 위해서 시스템의 많은 부분을 전문 컴포넌트 개발 회사에서 컴포넌트를 구입하여 구현하게 될 것이다. 이것은 기업에서 사용되는 아웃소싱의 원리와 비슷하다.

조립에 사용된 컴포넌트의 성능은 응용프로그램의 성능에 영향을 미친다. 따라서 응용프로그램 개발자는 컴포넌트를 구매시 컴포넌트의 객관적 성능을 알아야 한다[11].

컴포넌트는 통합 응용프로그램의 성능에 영향을 미친다. 컴포넌트 기반 소프트웨어 개발 테스트는 기존의 테스트 방법과 다르다[3][9].

컴포넌트 기반 소프트웨어를 위한 테스트 방법이 컴포넌트의 특성을 고려하여 다른 것처럼 컴포넌트의 성능 측정 방법도 기존의 방식만으로는 적합하지 않다. 분산 응용 프로그램에 적합한 컴포넌트를 측정하기 위해서는 분산환경을 고려하지 않은 기존의 측정 방식외에 추가적인 측정 방법이 필요하다. 기존 성능측정방식은 바이너리 형태의 독립된 모듈단위인 컴포넌트 단위로 성능을 측정하기 보다는 응용프로그램 레벨에서 응용프로그램의 기능 및 제어들을 테스트하는 것이었다. 컴포넌트는 특정 컴포넌트 아키텍처를 기반으로 구현된 것이기 때문에 일반적으로 컴포넌트 아키텍처에 따라 측정방법이 다르게 된다[3][11][12].

현재 문헌에 나타난 컴포넌트의 성능 측정 방법에는 컴포넌트의 응답시간, 효율성(throughput), 메소드(method) 처리시간, 풀(pool)의 개수에 따른 예외 발생률등이 있다[6][7].

본 논문에서는 분산환경에서 컴포넌트의 성능을 측정하기 위한 추가적 요소들을 제안하고 그 요소들을 측정하기위한 시스템을 설계한다. 여기에는 각 빈들의 처리 응답시간, 트랜잭션의 응답시간, 풀(pool) 크기에 따른 CPU 사용률, 알고리즘 처리시간, 컴포넌트를 처리하

기 위해서 필요한 힙 크기등이 포함된다.

본 논문의 구성은 다음과 같다. 2장에서는 컴포넌트에 대한 개념 및 특징과 EJB을 설명한다. 3장에서는 기존의 컴폰넌트 및 자바 프로그램 성능 측정 방법에 대해서 설명한다. 4장에서는 본 논문에서 제안하는 컴포넌트의 성능측정방법을 제시한다. 5장에서는 4장에서 제시한 성능측정방법에 따른 시스템을 설계한다. 6장에서는 결론 및 향후 연구에 관해서 논한다.

2. 컴포넌트의 정의와 특징 그리고 EJB 아키텍처

객체지향 기술을 대규모 시스템 개발에 효과적으로 적용하기 위해 개발되고 있는 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. Rational 사의 Pilippe Krutchen은 '컴포넌트는 잘 정의된 아키텍처 상에서 어떠한 기능을 수행하는 시스템에 독립적이면서 대치 가능한 부분이다' 라고 정의한다. Gartner 그룹은 '컴포넌트는 동적으로 바인딩 할 수 있는 하나 이상의 프로그램을 하나의 단위로 패키지한 것으로 실행시간에 인터페이스를 통해서 접근한 것'이라고 정의한다. Szyperski는 '컴포넌트는 스펙에서 명시된 인터페이스들과 명확한 문맥 의존성을 가진 조합의 단위로서 독립적으로 보급될 수 있다.'라고 설명한다[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. 즉 컴포넌트를 만들기위해서는 컴포넌트를 사용할 아키텍쳐를 고려해서 만들어야 한다. 다섯째 컴포넌트는 커스터마이징(customizing)이 가능해야 한다. 여섯째, 컴포넌트는 그의 원래 내용으로부터 분리될 수 있어야 한다. 컴포넌트들은 다른 응용프로그램에 사용될 수 있다[2].
본 논문에서는 EJB 아키텍처 기반 컴포넌트의 성능 측정에 관해서 설명한다[4].

2.2 EJB 아키텍처

Sun사의 EJB 아키텍처는 컴포넌트 기반의 분산 응용프로그램의 개발과 배치를 위한 컴포넌트 아키텍처이다. EJB는 자바 API와 호환되며, 엔터프라이즈 자바빈과 자바가 아닌 다른 언어로 구현된 어플리케이션과의 상호 연동이 지원된다.

EJB에서는 빈을 기본적인 컴포넌트 단위로 정의하고 있다. 모든 EJB 빈은 클래스가 지원하는 기능과 상태, 자료의 특징에 따라 세션 빈(Session Bean)과 엔티티 빈(Entity Bean)으로 나뉘게 된다.

EJB 빈은 컨테이너 안에 설치된다. 클라이언트는 직접 EJB 빈을 사용할 수 없다. 클라이언트는 컨테이너에서 제공하는 홈 인터페이스(Home Interface)와 리모트 인터페이스(Remote Interface)를 이용하여 EJB 빈을 사용한다. 그림 1은 클라이언트로부터 EJB 서버에 있는 빈을 사용하는 흐름이다. 클라이언트가 빈을 사용하고자 할 때, 홈인터페이스로 인스턴스를 생성한다. 그리고 리모트인터페이스를 통해서 생성된 빈의 비즈니스 로직

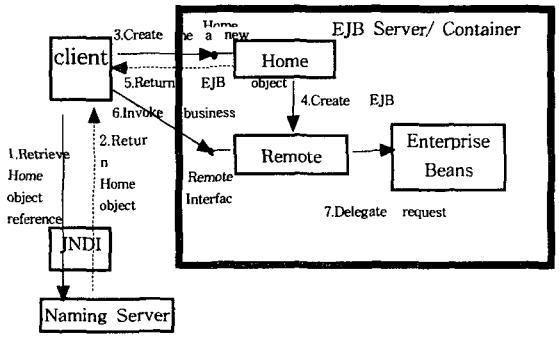

그림 1 클라이언트로부터 빈 사용하는 법 [10]

을 수행한다[5].

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

문헌에 언급된 기존의 성능측정방법에는 효율성, 빈의 처리응답시간, 예외발생률, 메소드 처리 응답 시간등이 있다[6][7].

효율성(throughput)

두 번째 빈의 평균 처리응답시간에 대한 역이다. 즉, 접속한 클라이언트 수에 따른 컴포넌트가 초당 처리하는 시간이다. 클라이언트에서 요구한 것을 주어진 시간 동안 컴포넌트가 할 수 있는 일의 양을 측정한다.

처리응답시간(Response time)

이 측정방법은 빈 하나를 생성하고, 그 빈을 수행하는데 걸리는 총 시간이다. 다시 말하면 EJB 아키텍처는 클라이언트가 직접 빈에 접근할 수 없도록 되어있다. 클라이언트는 홈 인터페이스를 통해서 빈을 하나 생성시킨 다음 리모트 인터페이스를 통해서 비즈니스 로직을 수행한다.

예외 발생률

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.

메소드 처리 응답 시간

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. 메소드의 종류별로 걸리는 시간을 측정한다[6][7].

3.2 자바 프로그램 성능 측정 방법

EJB는 자바 기반의 아키텍처를 가지기 때문에 자바 프로그램의 성능 측정방법을 이용할 수 있다.

첫째, CPU 측면이다. 사용빈도가 높은 쓰레드를 찾거나, CPU 사용률을 측정한다. CPU 안에서 동작하는 쓰레드 수등이다. 둘째, Memory 측면이다. 클래스들(Classes)과 인스턴스들이 할당받은 메모리를 측정한다. 셋째, 메모리 낭비(memory leak)를 찾고 제거한다. 넷째, 자바 프로그램 안에서 성능 병목현상이 발생한 곳을 발견한다[8].

4. 제안하는 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 두 개이상의 빈들을 조합해서 처리를 한다. 현재 처리시간(Response time) 측정방법은 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 처리 시간을 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 CPU 활용률이다. 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. 시스템 자원중 CPU 활용도를 측정한다. CPU가 메소드 처리들을 백분율로 나타낸다.

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 컴포넌트 성능 측정 방법 시스템 설계

컴포넌트의 성능측정방법을 지원하기위한 시스템을 설계한다.

5.1 컴포넌트 성능 측정 시스템의 아키텍처

EJB 컴포넌트는 재사용을 위해서 배치 서술자에 의해서 인터페이스를 표현한다. 이것은 화이트박스(WhiteBox) 해당하는 부분이다. 화이트박스부분을 사용하여 컴포넌트에서 사용하는 메소드 및 변수를 사용하여 클라이언트를 구성한다. 그림 2는 컴포넌트 성능측정 시스템을 위한 아키텍처이다.

Display : 클라이언트 모니터링 결과를 출력.

Monitoring Client : 클라이언트 소스 정보에 필요한 정보들을 모니터링하기위해 첨가된 부분.

MethodTimerClass(MCT) : 메소드 처리시간 정보를

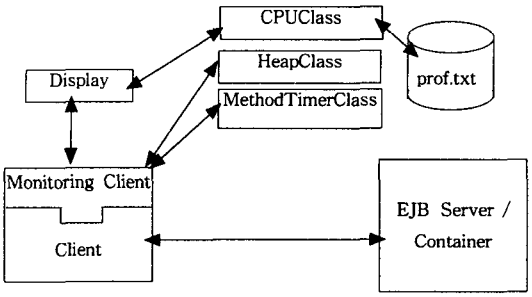

그림 2 컴포넌트 성능 측정 시스템 아키텍처

알기위한 클래스

HeapClass(HC) : 힙 정보를 알기위한 클래스

CPUClass(CC) : 자원 정보를 알기위한 클래스

prof.txt : 전처리 결과를 저장하는 파일

5.2 모듈디자인

처리응답시간은 메소드 처리시간을 이용한다. 처리시간에는 세 종류가 있다. 각 빈들의 처리 응답시간, 트랜잭션의 응답시간, 컴포넌트 알고리즘 처리시간이다. 각 빈들 처리시간은 해당 빈들에따른 메소드를 구별한다. 각 빈들의 메소드들 처리시간을 구하여 빈 종류에 따라 합한다. 트랜잭션 처리시간은 트랜잭션에 수행하는 빈여부를 구분하다. 트랜잭션에 참여하는 빈은 엔티티빈과 상태유지빈이다. 무상태빈은 트랜잭션 처리를 하지 않는다. 트랜잭션 메소드들 처리시간을 구하여 더한다. 알고리즘 처리시간은 초기처리하는 시간을 제외한다. 즉 리모트인터페이스를 통해서 처리하는 부분의 시간을 구한다.

그림 3 MTC는 자바의 System 클래스를 이용해서 메소드 처리시간을 구한다. 그림 4 HC는 자바의 Runtime 클래스를 이용하여 힙 크기를 구한다. 그림 5 CC는 자바의 Xrunhprof 옵션을 사용하여 전처리결과를 구하여 처리한다.

```
class MethodTimerClass {
   static String BeanName;
  static String MethName;
   int Number ;
   long Cumulation;
   long StartTime;
   long ProcessingTime;
   void MethodTime(){}
   void TimerStart(){}
   void TimerEnd(){}
   float AveTime(){}
 }
```

그림 3 MTC

```
class CPUClass{
 int Poolsize;
 int CPUUtil;
 int rank;
  Runtime R;
  ResourceClass(){ }
  long Hprof(){ }
  long FileOpen(){}
 string Search(String Wd)
 void Store(String data)
 void Rangking(Sting Mn)
}
```

그림 4 HC

```
class HeapClass{
  long Total, Free ;
  Runtime R;
  HeapClass(){}
  long FreeHeap(){}
  long UsedHeap(){}
  long TotalHeap(){}
}
```

그림 5 CC

6. 결론 및 향후 연구

컴포넌트의 사용이 증가함에 따라 컴포넌트의 개발자와 사용자는 다른 역할을 담당하게된다. 사용자들이 컴포넌트를 선택하기 위해서는 각 컴포넌트 성능을 비교하기 위한 방법이 요구된다. 본 논문은 EJB 컴포넌트의 성능 측정 방법을 제안하고, 제안한 컴포넌트 성능 측정 방법을 지원하는 성능 측정 도구를 설계하였다. 향후 연구에는 설계된 시스템을 구현할 예정이다.

7. 참고 문헌

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

[2] 박정희, "컴포넌트 기반의 소프트웨어 개발 및 사례 연구", 연세대학교 산업대학원 석사논문, 1999

[3] Jerry Ga, "Component testability and component testing challenges" , Software Engineering, Carnegie Mellon University, 2000

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

[5]Richard Monson-Haefel, "Enterprise Java Beans", pp.216-256, O'REILLY, 1999

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

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

[8] "JProbe Profiler V2.5", URL http://www.klgroup.com/jprobe/profiler/index.html, KL Group Inc.

[9] 최병주, "EJB컴포넌트의 맞춤테스트기법", pp.93-102, 소프트웨어공학기술 워크샵 2000 발표집, 1권1호 8.2000

[10] Ed Roman, "Mastering Enterprise JavaBeans and the Java2 Platform", Enterprise Edition, pp93, WILEY, 1999

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

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