* 본 연구는 정보통신부의 대학 S/W 연구센터 지원사업에 의해 수행된 것임.

EJB 컨테이너 기반 우선순위 리소스 풀링 서비스

이 준 규,° 김 동 호, 김 상 경, 안 순 신
고려대학교 전자공학과 네트워크 연구실
{mani99, kdh, skkim, sunshin}@dsys.korea.ac.kr

Resource Pooling Service with Priority in EJB Container

Jungyoo Lee,° Dongho Kim, Sangkyung Kim, Sunshin An
Computer Network Lab., Dept. Of Electronic Eng., Korea University

요 약

인터넷과 전자상거래 등이 보편화 됨에 따라 다수의 사용자 요구를 동시에 효율적으로 처리하며, 다양한 클라이언트/서버 요청을 신속하게 처리할 수 있는 다계층 구조의 분산처리 플랫폼에 대한 연구가 여러 연구 그룹에서 수행되고 있다. 이러한 연구 중의 하나가 자바를 이용한 Enterprise JavaBeans(EJB) [1]이다. EJB 는 de facto 표준으로 규격화되어 있다.

본 논문에서는 EJB 컨테이너의 리소스 풀링(Resource Pooling) 서비스에 우선순위 기능을 적용하여 그 성능을 개선하는 방법을 제시한다. 우선순위 처리 기능을 갖는 리소스 풀링 서비스를 설계하고 시뮬레이션을 통하여 성능을 분석한다.

1. 서론

1990 년대 이후로 인터넷은 급속히 확산되었으며 특히 WWW(World Wide Web), 전자상거래 등에 관련된 응용 프로그램등의 등장으로 인하여, 다수의 사용자를 수용할 수 있는 서버측 솔루션에 많은 관심이 집중되고 있다. 기존의 1 계층이나 2 계층 시스템은 기능을 추가할 때마다 서버가 비대해지는 현상이 발생하며, 사용자 수가 많아짐에 따라 서버 부하가 기하 급수적으로 커지게 되고, 시스템 관리 차원에서 많은 문제점을 갖고 있다. 3 계층 시스템은 이러한 문제점을 해결하기 위해 서비스를 분산하여 수행하므로 대용량, 대규모 비즈니스 응용에 적합하다. J2EE(Java 2, Enterprise Edition) [2]는 자바를 이용한 3 계층 시스템을 구축하는 기술이다. J2EE 는 여러가지 기술들로 이루어져 있는데, 그 중 EJB 는 자바 기반의 기술 중 가장 최근에 발전하고 있고, EJB 컴포넌트는 다양한 클라이언트/서버 환경간에 통신이 가능하다. EJB 표준안에 제시한 EJB 컴포넌트는 모두 같은 중요성을 가지고 동작한다. 어떤 하나의 컴포넌트가 선행되어 수행되지 않고, 모든 컴포넌트들이 서버에 들어온 순서대로 실행되고, 종료된다.

EJB 컴포넌트를 사용하여 은행의 전산시스템을 구성하는데 고객의 재무상태를 보여주는 빈과 고객의 입출금을 관리하는 빈이 있다고 하자. 여러 고객이 두 가지의 서비스를 동시에 요청하면, 고객의 입 출금을 처리하는 빈을 우선시하여 처리해야 하는 경우가 있다. 현재의 EJB 설계로는 이런 작업을 수행할 수 없다. 이에 본 논문에서는 엔터프라이즈 빈에 우선순위를 두어 처리하는 것이 필요하다는 것을 인식하고, 그 방법을 제시한다.

본 논문은 다음과 같이 구성되어 있다. 2 장에서는 리소스 풀링(Resource Pooling) [3]과 적용할 우선순위[4] 정책에 관하여 살펴보고, 3 장에서는 우선순위기능이 추가된 EJB 컨테이너의 모델을 제안한다. 4 장에서 실험 및 결과를 분석하고, 5 장에서 결론 및 향후연구를 기술하며 끝을 맺는다.

2. 리소스 풀링 서비스

비즈니스 시스템은 몇 천, 몇 만개의 객체를 가지면서, 많은 양의 작업을 수행하고 다수의 클라이언트의 접속을 유지해야 한다. 클라이언트의 접속하는 수가 증가할수록, 리소스의 유효량은 감소하고, 어느 시점에서는 수행 속도가 현저히 감소할 것이다. 이러한 것의 해결방안으로 EJB 컨테이너는 기본적으로 리소스 풀링 기능을 제공한다. 리소스 풀링의 개념은 이미 시스템 내의 비즈니스 객체와 데이터베이스의 접속을 분할 처리하기 위해 데이터베이스 접속 풀링(Database connection pooling) [5]이라는 기술로 널리 사용되고 있었다. 이 기술은 연결 수와 리소스 소비를 줄이고, 수행속도를 향상시켰다. 데이터베이스를 여러 번 새로이 접속하고 끊는 것 보다 한번 접속하고 풀링 상태로 있다가 재사용하는 것이 적은 양을 소비한다는 점에서 착안된 것이다. 이 기술을 클라이언트와 서버 안에서 사용되는 컴포넌트 간에 관계로 적용시킨 것이 리소스 풀링 혹은 인스턴스 풀링(instance pooling)이다. 리소스 풀링은 클라이언트의 요구에 대한 서비스를 수행하는데 컴포넌트 인스턴스와 리소스의 수를 줄일 수 있다. EJB 클라이언트는 실질적인 빈에 바로 접근할 수 없고, 리모트 인터페이스를 통해

연결되고, 엔터프라이즈 빈 인스턴스를 감싸고 있는 인터페이스를 통해서 동작을 한다. 이렇듯 리소스 풀링은 엔터프라이즈 빈에 간접적으로 접근할 경우에 더 나은 효력을 발휘한다.

3. 우선순위기능이 추가된 EJB 컨테이너의 리소스 풀링 설계

엔터프라이즈 빈에 우선순위기능을 부여하기 위해서는 두 가지 처리 방법을 고려할 수 있다. 하나의 방법은 EJB 컨테이너의 내부에서 리소스 풀링의 기능을 수행하는 과정 중에 EJB 컴포넌트들 간에 우선순위를 처리할 수 있다. 그리고 다른 방법은 EJB 컨테이너와 컴포넌트의 사이에 우선순위에 관한 처리를 하는 대리인(agent)을 두어서, 요구가 들어오는 컴포넌트들 간의 우선순위를 조사하고 스케줄링 작업을 한 후에 컨테이너에 수행을 의뢰하는 방법이 있다. 본 논문에서는 첫번째 우선순위기능을 추가하는 방법을 선택해서 적용할 방안을 기술한다.

우선순위를 부여하기 위해서는 두 부분에 나누어서 처리를 해야 한다.

- 엔터프라이즈 빈 클래스에 우선순위를 나타내는 정보를 추가해야 한다.
- 인스턴스를 관리하는 리소스 풀링 기능에 우선순위를 처리하는 루틴을 추가한다.

이에 대한 자세한 사항은 아래에 기술한다.

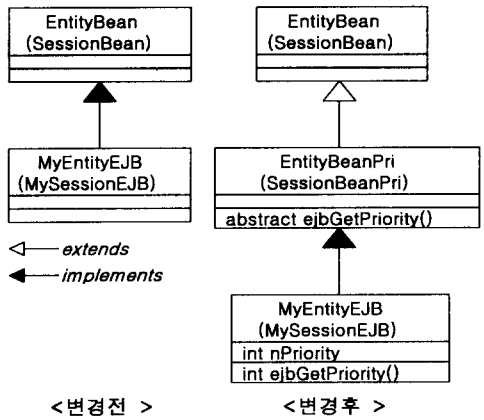
3.1 엔터프라이즈 빈 클래스 처리

엔터프라이즈 빈의 클래스에 우선순위기능을 추가하기 위해서는 상속관계를 변경해야 한다. 그 방법은 다음과 같다.

- EntityBean, SessionBean 의 인터페이스를 상속받는 EntityBeanPri, SessionBeanPri 을 작성한다.
- EntityBeanPri, SessionBeanPri 의 인터페이스에 public abstract int ejbGetPriority()의 가상함수를 추가한다.
- 기존의 EntityBean, SessionBean 의 인터페이스를 구현하는 클래스(MyEntityEJB, MySessionEJB)의 implements 를 EntityBeanPri, SessionBeanPri 로 변환한다.
- 인터페이스 구현부인 MyEntity, MySession 의 코드를 아래와 같이 변경한다.

```
public class MyEntityEJB(혹은 MySessionEJB)
  implements   EntityBeanPri, (혹은 SessionBeanPri) {
    private int nPriority;
≪≪≪≪≪ 중략 (다른 기능들) ≫≫≫≫≫
    public void ejbCreate()
    { nPriority = 4; }  //사용자가 값(1-r)을 정한다.
    public int ejbGetPriority()
    { return nPriority;}
}
```

이때, 우선순위를 두지 않고 일반적으로 사용할 경우에는 인터페이스 구현부에서 ejbGetPriority 함수에 대한 처리를 하지 않으면 일반적인 엔터프라이즈 빈의 동작을 한다. 이상과 같이 코드를 변환시키면 엔터프라이즈 빈에 새로운 nPriority 필드를 통해서 우선순위를 부여할 수 있다. 변경 전, 후의 클래스 계층도를 보면 [그림 1]과 같다.



[그림 1] 설계방안 적용 전후의 클래스 계층도

3.2 리소스 풀링 처리 변환

인스턴스를 관리하는 리소스 풀링 기능에 우선순위를 처리하는 루틴을 추가해야 한다. 각 EJB 컨테이너에는 엔터프라이즈 빈을 읽고 인스턴스를 부여하는 리소스 풀링 기능을 처리하는 루틴이 있다. 엔터프라이즈 빈의 리소스 풀링 기본 동작은 다음과 같다.

① 서버의 EJB Home 과의 연결을 위해 Home Stub 을 생성한다.
② EJB Home 은 EJB Object 를 인스턴스한다. 이때 EJB Object 는 풀 상태이다.
③ EJB Object 에 하나의 인스턴스를 할당한다.
④ EJB Object 의 remote reference 를 클라이언트에 전송한다.
⑤ 클라이언트의 사용이 끝난 후에는 인스턴스는 풀에 다시 저장한다.

상기에 제시된 방법은 매번 새로운 인스턴스를 생성하거나 소멸시킬 필요 없이 이미 생성된 인스턴스는 재사용을 하여 효율을 높일 수 있다. 여기에 ③번의 인스턴스를 할당하는 과정에서 우선순위가 높은 EJB Object 부터 처리를 하도록 해야 한다.

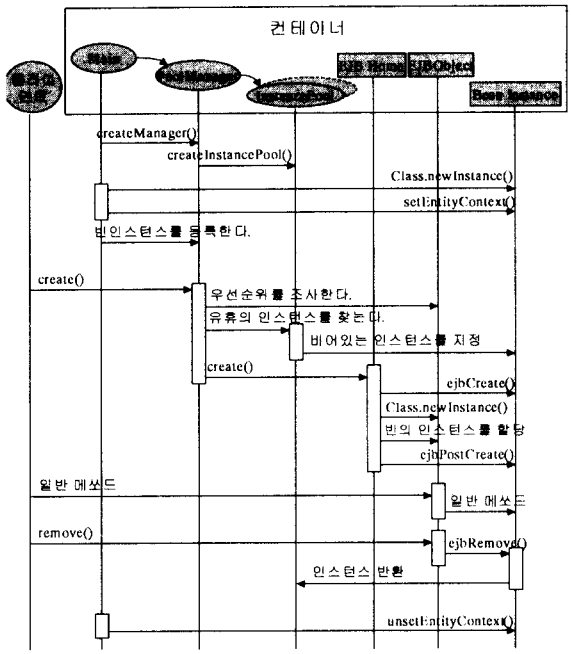
새로운 기능을 추가하는 설계도에서, 컨테이너 안에는 컨테이너 프로그램의 Main 이 있고, Pool Manager, Instance Pool, EJB Home, EJB Object, Bean Instance 등이 있다. [그림 2]는 엔터프라이즈 빈의 리소스 풀링을 수행하기 위해서 각각의 개체와 클라이언트 간의 상호작용을 나타낸 시퀀스 다이어그램이다. 여기서 중요한 역할을 하는 Pool Manager 와 InstancePool 가 제공하는 기능은 다음과 같다.

Pool Manager 클래스

- 시스템에 있는 모든 EJB 컴포넌트를 로드(load)하고 등록한다.
- Instance Pool 를 생성하고, EJB 컴포넌트와 매핑시킨다.
- 우선순위에 따라 클라이언트의 요구에 맞는 인스턴스와 연결을 시킨다.

Instance Pool 클래스

- 풀(Pool) 상태의 인스턴스를 얻어낸다.
- 사용이 종료된 인스턴스를 풀에 반환한다.
- 프로그램 종료시에 풀의 상태에 있는 모든 인스턴스와 리소스를 시스템에 반환한다.

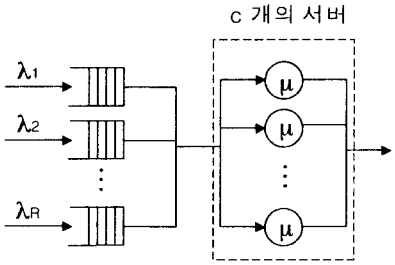


[그림 2] 엔터프라이즈 빈의 리소스 풀링

4. 시뮬레이션

4.1 시스템 모델

우선순위가 적용된 리소스 풀링을 수학적으로 모델링 한다면 여러 개의 서버(c)를 갖고 있는 [그림 3]과 같이 표현할 수 있다. 서비스를 요청하는 Call 은 클래스 i (i = 1, 2,....., R)을 갖고 λ_i 의 비율의 지수분포로 도착하고, 서비스 시간은 μ의 지수분포를 갖는다고 가정하자. 이때 i 값이 커질수록 우선순위가 높다고 가정한다. $\rho_i = \lambda_i / \mu$ 이고, W_i 는 서비스를 받기까지 기다리는 시간의 평균이다.



[그림 3] 우선순위 M / M / c 시스템[6]

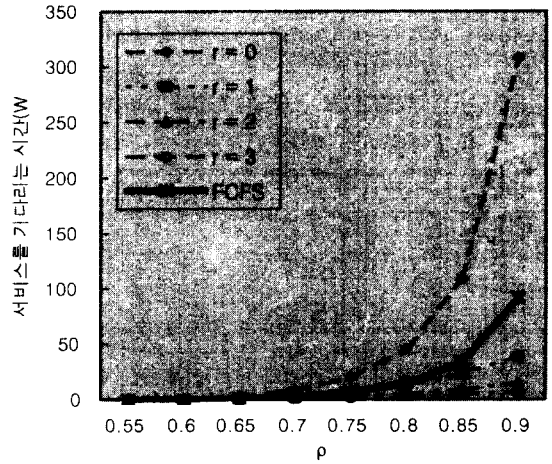
4.2 시뮬레이션 환경

사용언어 : C++ 프로그램
우선순위 M / M / m 시스템 :

- 입력 : 포아송 과정
- 서버 수(m) : 20 개
- 우선순위 클래스 계층 수(R) : 4 단계
- 서비스 수행 속도 : 지수 분포
- 전체 서비스 셀의 수 : 100,000 개
  - 수렴 지점 : 70,000 개
- 각 클래스에 해당하는 셀의 개수 : 랜덤

4.3 결과 분석

[그림 4]의 시뮬레이션 결과를 보면, 우선순위가 높은 컴포넌트(r = 3, 2, 1; 비율: 70%정도)는 FCFS 처리를 하는 것보다 월등히 적은 시간 안에 서비스를 받는다. 특히 서버의 부하가 커지면 커질수록 중요한 컴포넌트들은 선행되어 빠르게 서비스를 받는 장점이 있다. 그런데 일부 우선순위가 낮은 컴포넌트(r = 0)는 기다리는 시간이 길어진다는 단점이 있다.



[그림 4] FCFS 와 우선순위 비교

5. 결론 및 향후 연구

본 논문에서는 EJB 컨테이너에서 컴포넌트의 일괄적인 처리방식을, 중요한 컴포넌트를 우선시하여 처리하는 방식으로 변환하는 방법을 설계하였고, 우선순위가 높은 클래스를 갖는 컴포넌트는 FCFS 로 설계한 것에 비해 빠르게 처리된다는 것을 보였다. 이것을 서버의 컴포넌트를 배치할 때 적용하면 사용자와 서비스 제공자에게 중요한 업무처리를 빠른 시간 안에 효과적으로 처리할 수 있다.

본 논문에서는 우선순위가 높은 클래스에 대해서는 서비스를 받기까지 기다리는 시간이 작았지만, 상대적으로 낮은 클래스는 많은 시간을 기다려야 하는 문제점이 존재한다. 향후 좀더 효율적인 우선순위 정책을 제공하는 전체 시스템 성능향상 방안이 연구되어야 한다.

참고 문헌

[1] Vlada Matena & Mark Hapner, *"Enterprise JavaBeans™ specification v1.1"*, Sun Microsystems, 1998.

[2] *"Designing Enterprise Applications with the Java™ 2 Platform, Enterprise Edition(version 1.0)"*, Sun Microsystems, 2000.

[3] Bolch, Greiner, de Meer, and Trivedi, *"Professional Java Server Programming"*, WROX, 1998.

[4] Richard Monson-Haefel, *"Enterprise Java Beans"*, O'REILLY, 1999.

[5] Bolch, Greiner, de Meer, and Trivedi, *"Queueing Networks and Markov Chains"*, JOHN WILEY & SONS, 1998.

[6] 이호우, *"대기행렬이론"*, 시그마프레스, 1998.