JACE 그룹통신시스템을 이용한 신뢰성있는 공유객체공간의 개발†

안건태° · 문남두 · 정현락 · 유양우 · 이명준
울산대학교 컴퓨터 · 정보통신공학부

Development of Reliable Object Space using JACE group communication system

Geon-Tae Ahn° · Nam-Doo Moon · Hyun-Rak Jung · Yang-Woo Yu · Myung-Joon Lee
School of Computer Engineering · Information Technology, Univ. of Ulsan.

요 약

인터넷과 네트웍 기술의 급속한 성장에 따라 간단하고 효율적이며, 관리가 용이한 분산 어플리케이션의 필요성이 증대되었다. 이러한 요구사항을 만족시키기 위하여 네트워크 상에 분산되어 있는 자원, 서비스 및 객체의 공유와 이들 사이의 통신을 일관된 방법으로 지원하는 튜플스페이스(Tuple space) 시스템들이 제안되었다. 그러나, 기존의 시스템들은 서버를 통한 중앙 집중적인 공유공간(shared space)구조를 가지고 있으므로 서버의 실패(crash)나 네트워크의 분할(partition)과 같은 전산망의 결함이 발생하였을 경우 지속적인 서비스를 제공할 수 없게 된다.

본 논문에서는 특정 목적에 따라 그룹화된 공유공간의 생성이 가능하고 객체를 관련노드에 중복(replication)해서 저장할 수 있는 튜플스페이스 시스템인 ObjectSpace의 설계 및 구현에 대하여 설명한다. ObjectSpace는 객체를 공유하기 위하여 공유공간에 객체를 저장하는 기능과 공유된 객체를 조회하는 기능을 제공한다. 중복된 객체들에 대하여 일관성을 유지하기 위하여 프로세스 그룹을 지원하는 JACE 그룹통신시스템을 하부구조로 사용하여 구현하였다.

1. 서 론

오늘날 대부분의 컴퓨팅 환경은 지역적인 네트워크를 사이에 두고 서로 상호 작용하는 멀티 시스템 환경을 이루고 있다. 이러한 분산 환경 하에서는 공동작업을 수행하기 위해서는 자원, 서비스 및 객체들에 대한 공유나 이들 사이의 통신을 간단하고 일관된 방법으로 제공하여 주는 것이 바람직하다. 이러한 요구사항을 만족시키기 위하여 전역적인 공유공간(global shared space)을 제공하는 튜플스페이스 시스템들이 제안되었다. 튜플스페이스 시스템은 프로그래밍 인터페이스가 간단하고 사용자들이 쉽게 이용할 수 있는 튜플스페이스라는 공유공간을 제공함으로써 네트웍 상에 분산된 자원이나 객체들에 대한 효율적인 관리 방법을 제공하고 있다. 튜플스페이스가 저장하고 관리하는 단위를 튜플(tuple)이라고 하며 모든 연산은 이 튜플을 기본단위로 수행된다. 최근에는 자바언어를 이용하여 튜플스페이스를 구현한 시스템들이 나타났는데 그 대표적인 예가 썬 마이크로시스템즈사의 JavaSpace[5]가 있다. JavaSpace는 튜플대신 객체를 기본 연산 단위로 사용함으로써 데이터 뿐 만 아니라 행동방식(behavior)도 스페이스 내에 저장 가능하다.

기존의 시스템들은 서버를 통한 중앙 집중적인 튜플스페이스 구조를 가지고 있으므로 서버의 실패나 네트워크의 분할과 같은 전산망의 결함이 발생하였을 경우 지속적인 서비스를 제공할 수 없게 된다. 따라서, 본 연구에서는 객체를 관련노드에 중복해서 저장할 수 있는 튜플스페이스 시스템인 ObjectSpace 시스템을 개발하였다. 하부 구조로 프로세스그룹을 지원하는 JACE(Java Advanced Communication Environments)그룹통신시스템[1][2]을 이용하였다. JACE는 응용서비스들을 프로세스그룹으로 동작할 수 있도록 지원하며, 네트워크 분할 및 서버의 실패와 같은 전산망의 결함에도 불구하고 자바응용서비스들이 안정적이고 지속적인 서비스를 제공할 수 있도록 그룹 구성원 사이의 일관성을 지원한다.

본 논문의 구성은 다음과 같다. 2장은 관련연구로서 기존 튜플스페이스 시스템에 대하여 기술한다. 3장에서는 ObjectSpace 시스템의 전반적인 구조와 각 모듈의 역할에 대하여 설명한다. 4장에서는 ObjectSpace를 이용하여 분산응용프로그램을 개발하기 위한 방법에 관하여 기술한다. 5장에서 결론 및 향후 연구방향에 대해 살펴본다.

†본 연구는 '99년도 정보통신연구진흥원 사업과제의 지원으로 수행되었음.

2. 관련 연구

2.1 린다(Linda)

린다[3][4]는 병행처리를 지원하기 위한 새로운 모델로 제안되었다. 기본 원리는 각 프로세스들이 튜플이라는 기본 단위를 위한 저장소 역할을 하는 튜플스페이스를 공유하는 것이다. 각 튜플은 통신을 위한 실제 정보를 포함하는 필드(field)들의 모임이다. 튜플스페이스는 튜플들의 집합을 나타내며 여러 프로세스들에 의하여 동시적으로 접근되어질 수 있다. 린다는 튜플스페이스에 대하여 4가지의 기본연산을 제공한다. 프로세스들은 다음 연산을 이용하여 튜플스페이스를 통하여 정보교환 및 통신을 하게 된다.

1) rd(t)

튜플스페이스에서 튜플 t를 읽어 오는 연산을 수행한다.

2) in(t)
rd()와 비슷한 기능을 수행하나 읽은 후
튜플스페이스로 부터 튜플 t를 삭제한다.

3) out(t)
튜플스페이스에 튜플 t를 저장한다.

4) eval(expression)
인자로 독립적으로 수행되는 새로운 프로세스들을 생성하여 계산된 최종결과를 튜플 스페이스에 저장하게 된다.

2.2 자바스페이스(JavaSpace)

자바스페이스는 인터넷환경에서 분산어플리케이션을 개발하는데 있어서 간단하면서 쉬운 해결책을 제시해 주는 기술이다. 자바스페이스는 튜플스페이스와 유사한데, 기본 저장단위가 튜플이 아니라 자바 클래스 객체인 엔트리(entry)를 이용한다는 차이점이 있다. 자바스페이스 기술은 자바로 구현되었기 때문에 엔트리가 가지는 타입(type)이나 값들에 대하여 자바프로그래밍 언어의 강력한(strong) 타입시스템(type system)을 이용할 수 있는 장점을 가지고 있다. 또한 외부 객체들에게 자바스페이스의 내용이 변할 때마다 알려주는 기능도 제공한다. 자바스페이스는 다음과 같은 간단하면서도 강력한 기본연산을 제공함으로써 객체에 대한 공유와 이들 사이의 통신을 위한 일관된 환경을 지원해주고 있다. .

1) write
자바스페이스에 주어진 엔트리를 저장한다.

2) read
주어진 템플릿과 일치하는 엔트리를 자바스페이스로부터 읽어온다.

3) take
자바스페이스에 엔트리를 읽은 후 제거한다.

4) notify
특정 엔트리가 자바스페이스에 저장될 때 알려주는 기능을 수행한다.

3. ObjectSpace의 구조

공유객체공간은 분산환경에서 자원의 효율적인 공유와 통신을 위한 간단하면서 통일된 방법을 제시해 주는 신 개념이다. 본 논문에서는 공유객체공간의 신뢰성을 높일 수 있는 방법으로 객체를 관련노드에 중복(replication)해서 저장할 수 있는 튜플스페이스 시스템인 ObjectSpace를 개발하였다. 하부 구조로 사용된 JACE 시스템은 기본적으로 네트워크의 분할을 고려한 시스템으로 자바 응용서비스들을 프로세스그룹의 형태로 동작할 수 있도록 지원한다.

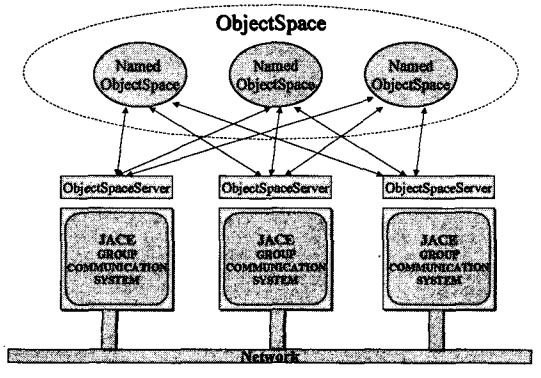

그림 1 ObjectSpace 구조

3장에서는 본 시스템의 세부사항을 설명할 것이다. 그림 1은 ObjectSpace의 전체구조를 나타낸다. 각 노드에서는 그룹통신시스템인 JACE 시스템과 ObjectSpace의 서버(server) 역할을 하는 ObjectSpaceServer가 동작하게 된다. 동일한 이름 하에 그룹화 지어진 공유객체공간들은 객체 중복과정을 통해서 서로 일관성을 유지하게 되며 클라이어트측에서 그룹을 생성할 때마다 관련 노드에 동일한 이름의 공유객체공간이 생성되게 된다. 공유객체공간이 관리하는 기본단위인 엔트리(Entry)는 다음과 같다.

3.1 엔트리(Entry)의 구조

ObjectSpace를 통해 관리되는 기본 단위는 엔트리(Entry)라고 하는 자바 언어의 serializable 인터페이스를 구현(implements)한 추상클래스이다. 다음은 엔트리 구조를 나타낸 것이다.

```
public abstract class Entry implements Serializable
{
      String name;
      Object value;

      public Entry(String aName, Object aValue){
          name = aName;
          value = aValue;
      }
      public Entry(String aName){ name = aName;  }
      public void setName(String aName){ name =
          aName; }
      public String getName(){ return name; }
      public void setValue(Object aValue){ value =
          aValue; }
      public Object getValue(){ return value; }

      public static boolean equals(Entry e1, Entry
          e2) { return e1.equals(e2); }
      public static int hashCode(Entry entry) {
          return entry.hashCode(); }
      public static String toString(Entry entry) {
          return entry.toString(); }
}
```

응용프로그램을 개발하는 개발자들은 이 추상클래스를 상속(inheritance)하여 자신들이 ObjectSpace를 통하여 공유하기를 원하는 객체들을 직접 설계 할 수 있다.

3.2 ObjectSpace 기본API

ObjectSpace가 제공하는 기본 API로는 read, write, 그리고 take가 있는데 그 기능은 린다 언어와 유사하다. 각각 두 가지 형태의 메소드(method)를 제공하고 있는데, 각 메소드의 인자로는 사용하고자 하는 공유객체공간의 이름을 직접 지정할 수도 있고 그룹 이름없이 객체만을 인자로 사용할 수도 있다. 후자의 경우는 기본 그룹으로 지정 객체를 저장하게 된다.

- 공유객체공간의 생성 및 제거

특정한 이름 하에 공유객체공간을 만들 수 있는데 같은 이름으로 참여하는 모든 응용프로세스들은 그룹으로 동작될 수 있다. createGroup(String aGroup)은 주어진 인자를 이름으로하여 ObjectSpace를 생성하고 createJoinGroup(String aGroup)은 주어진 인자를 이름으로하는 ObjectSpace가 없어면 하나 만들고 있으면 참여(join)하게 된다. removeGroup(String aGroup) ObjectSpace를 삭제하는 연산이다.

- 공유객체공간에 객체 저장하기와 읽기

린다(Linda)에서 제공하는 read, write, 그리고 take를 기본함수로 제공해 준다. 그 기능은 린다와 유사하며, read()는 공유객체공간으로부터 객체를 읽어 오는 것이고, write()는 공유객체공간에 객체

하나를 저장하는 연산이며, 마지막으로 take는 공유객체공간으로 부터 객체 하나를 읽어온 후 스페이스 내에서 그 객체를 삭제해 버린다. read 와 take에 대해서는 인자로 대기시간(wait-time)을 주어 원하는 객체가 공유객체공간에 없을 시 주어진 시간만큼 기다리게 할 수 있다. 그 인자 값이 NO-WAIT일 경우는 그 결과 값을 바로 리턴(return)하게 된다.

```
public interface ObjectSpaceClient {
  // Object Space 에 기본 그룹을 말한다
  final static String DEFAULT_GROUP =
                              "###_DEFAULT_GROUP_###";

  // Don't wait at all
  final static long NO_WAIT = -1;

  Entry read(String aGroup, Entry aEntry, long aTime) throws
     ObjectSpaceReadException ;
  Entry take(String aGroup, Entry aEntry, long aTime) throws
     ObjectSpaceTakeException ;

  Entry read(Entry aEntry) throws ObjectSpaceReadException ;
  void write(Entry aEntry) throws ObjectSpaceWriteException ;
  Entry take(Entry aEntry) throws ObjectSpaceTakeException ;

  void createJoinGroup(String aGroup) throws
     ObjectSpaceCJGException ;
  void createGroup(String aGroup) throws
     ObjectSpaceCGException ;
  void removeGroup(String aGroup) throws
     ObjectSpaceRMGException ;

}
```

위 의 코드는 ObjectSpace에서 제공하는 클라이언트측 인터페이스를 보여준다. ObjectSpace를 이용해 분산어플리케이션을 개발하기 위해서는 ObjectSpaceClient 인터페이스를 반드시 구현하여야 한다.

4. ObjectSpace를 이용한 분산어플리케이션 예

ObjectSpace를 이용해서 분산어플리케이션들을 개발할 경우 ObjectSpace가 가지는 그룹화 특성과 전산망의 결함에도 불구하고 안정적이고 지속적인 서비스를 제공해 주는 신뢰성 때문에 보다 견고한 시스템을 구성할 수 있다. 4장에서는 실제로 ObjectSpace를 이용하여 분산어플리케이션을 개발하기

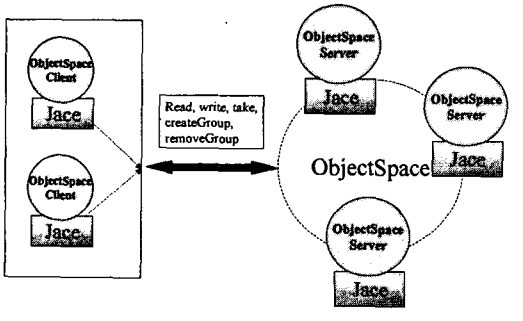

그림 2 그룹으로 동작하는 ObjectSpace

위한 예제를 보여준다. 그림2는 ObjectSpace가 동작하는 원리를 보여주고 있다.

ObjectSpace는 JACE를 기반구조로 가지고 있다. 따라서, 분산응용프로그램 개발자는 우선 ObjectSpaceClient 인터페이스를 구현한 ObjectSpaceClientImpl이라는 객체에 대한 참조를 얻고 난 후, 특정한 이름을 지정하여 ObjectSpace를 생성하게 된다. 그리고, 공유를 원하는 객체들을 생성하고, ObjectSpace가 제공하는 기본API를 이용하여 공유공간에 객체를 저장하거나 공유된 객체를 조회하는 연산을 이용함으로써 정보를 공유하고 통신할 수 있게 된다.

```
public class SampleApplication{
   public SampleApplication() {

      ObjectSpaceClientImpl ObjectSpace ;

      ObjectSpace = ObjectSpaceClientImpl.create();

      //"MyGroup"은 ObjectSpace명
      String group =  "MyGroup";

      //공유객체를 생성
      TestEntry myEntry1 = new TestEntry("Object1",
                    "Obj1_Val"));
      TestEntry myEntry2 = new TestEntry("Object1",
                          null));
      TestEntry repEntry1, repEntry2 ;
       //원하는 사용자 연산을 수행
      try{
        ObjectSpace.createGroup(group);
        ObjectSpace.write(group, myEntry1);
        repEntry1 =
        (TestEntry)ObjectSpace.read(group, myEntry2);
        repEntry2 =
        (TestEntry)ObjectSpace.take(group, myEntry2);
      }catch (ObjectSpaceException e){
             System.out.println("Error !! \n");
    }
}
```

5. 결론 및 향후과제

본 논문에서는 객체들을 관련 노드에 중복하여 저장할 수 있는 튜플스페이스 시스템인 ObjectSpace시스템의 설계와 구현에 대하여 기술하였다. 중복된 객체들의 일관성을 유지하기 위하여 프로세스그룹을 지원하는 JACE시스템을 이용하였다. 개발된 ObjectSpace 시스템을 이용함으로써 분산어플리케이션 개발자들은 간단하고 효율적이며, 관리가 용이한 프로그램을 개발할 수 있다.

향후 과제로는 객체들의 중복에 따른 오버헤드를 줄이고 ObjectSpace 시스템의 성능향상을 위하여 실제 객체들의 정보는 저장한 노드에서 유지하고 공유객체공간의 뷰(view)만을 공유할 수 있는 시스템을 개발할 예정이다.

[참고문헌]

[1] 최혁재 외, "자바응용서비스 개발을 위한 JACE프로그래밍 인터페이스." 한국정보과학회 '99 봄 학술발표논문집(26, 1), pp.382-384. 1999.

[2] 문남두, 최혁재, 유양우, 박양수, 이명준 "자바를 이용한 Extended Virtual Synchrony의 지원." 한국정보과학회 '98 봄 학술발표논문집 (25, 2), pp.409-411. 1998.

[3] N. Carriero, D. Gelernter, and J. Leichter, "Distributed Data Structures in Linda," proc. ACM Symp. Principles of Programming Languages, New Your: ACM, PP.236-242, 1986

[4] N. Carriero and D. Gelernter, "Linda in Context", Communications of the ACM32, No. 4, pp.444-458(April 1989)

[5] Sun Microsystems, Inc. "JavaSpace Technology", *"http://java.sun.com/products/javaspaces/index.html"*

[6] Antony I. T. Rowstron, Alan Wood: "An Efficient Distributed Tuple Space Implementation for Networks of Workstations" . Euro-Par, Vol. I 1996: PP. 510-513.