



Tehnologii Java

Curs -

Cristian Frăsinaru

`acf@infoiasi.ro`

Facultatea de Informatică

Universitatea "Al. I. Cuza" Iași



Programare distribuită

Cuprins

- Programare distribuită
- Arhitecturi și modele
- RMI
- JavaSpaces

Programare distribuită



Aplicație distribuită = mulțime de agenți care colaborează în vederea îndeplinirii unei anumite sarcini prin diverse mecanisme de coordonare și comunicare.

Agent = Entitate de calcul mai "inteligentă" decât un obiect, având un anumit grad de autonomie, capabilă să fie conștientă de sarcinile sale, de mediul înconjurător și de progresul activității sale.



Arhitecturi și modele

● Legătură

- Strâns legate (Tightly coupled)
- Slab legate (Loosely coupled)

● Organizare

- Client-server (N-tier)
- Peer-to-Peer
- Bazate pe servicii

● Comunicare

- Memorie partajata
- Transmitere de mesaje

Memorie partajată

Spațiu de tuple/obiecte



Metafora tablei

Implementări ale paradigmei de comunicare prin memorie partajată

- Spațiu de tuple
- Spațiu de obiecte

Caracteristici: acces concurent, excludere mutuală



Coordonare și comunicare

Accesul la un spațiu de tuple/obiecte presupune existența unui model de coordonare și comunicare. Un exemplu de astfel de model este **Linda**, ce definește un set de primitive atomice:

- **in** - citire cu eliminare
- **rd** - citire fără eliminare
- **out** - creează o tuplă și o scrie în spațiu
- **eval** - creează un proces de evaluare a unei tuple, scriind rezultatul în spațiu

Există implementări pentru Prolog, Ruby (Rinda), Python, C, Smalltalk, Java, Lisp.

JavaSpaces



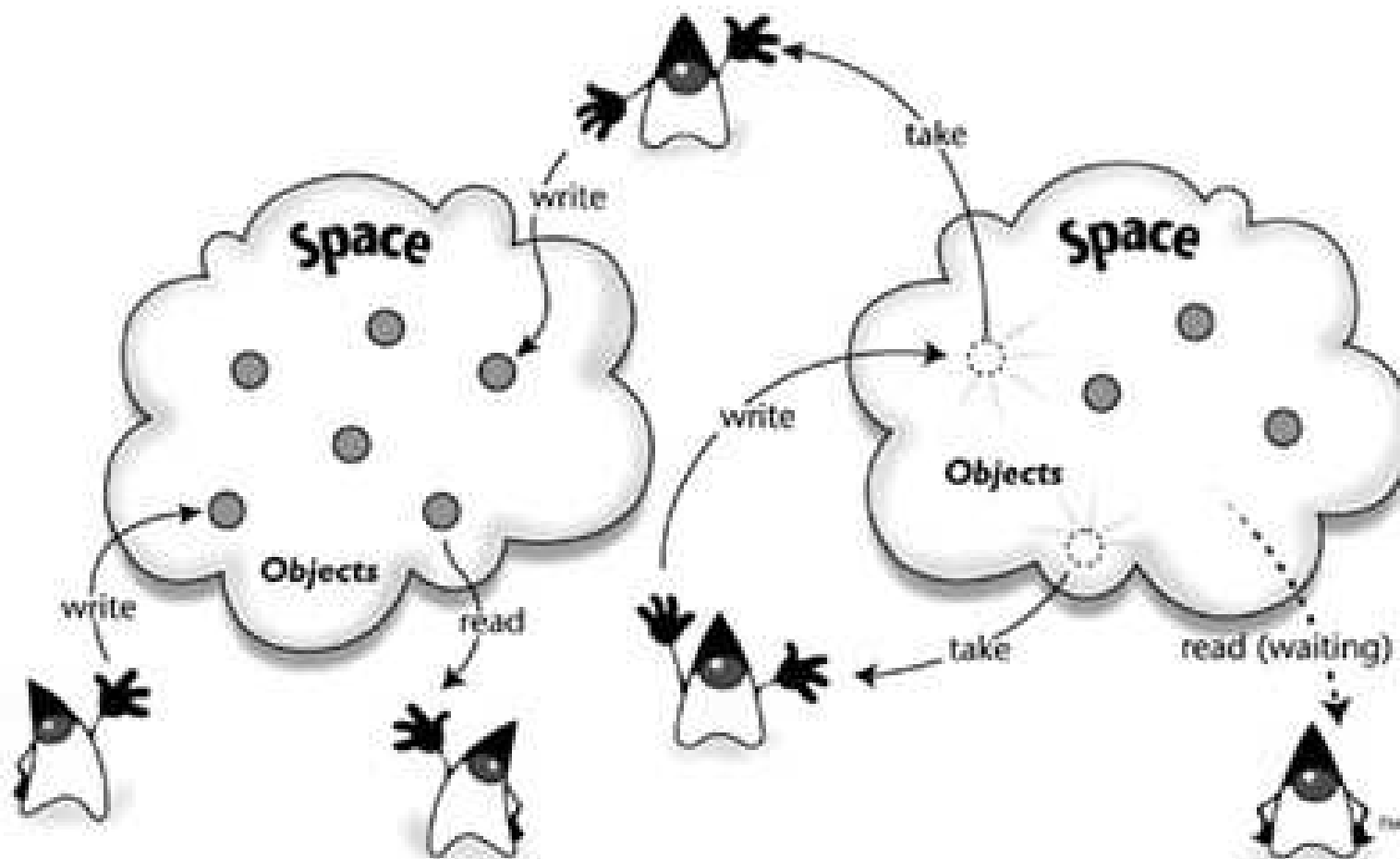
JavaSpaces is a service specification providing a distributed object exchange and coordination mechanism (which may or may not be persistent) for Java objects.

- Parte din proiectul Jini
- Aderă la modelul Linda
- Tehnică de obținere a scalabilității

Implementări: GigaSpaces, Blitz, etc.



Imagine de ansamblu



Caracteristici de bază

- Spaces are **shared**
memorii comune distribuite în rețea
- Spaces are **persistent**
obiectele sunt salvate pentru un timp sau indefinit
- Spaces are **associative**
cautarea obiectelor se face pe baza atributelor
(cautare asociativa)
- Spaces are **transactionally** secure
mai multe operații pot fi executate în mod atomic
- Spaces allow us to **exchange** executable content

Exemplul "Hello World"

- 1. Creăm o clasă care expune o anumită funcționalitate
- 2. Scriem obiecte din acea clasă într-un spațiu
- 3. Regăsim obiecte pe baza proprietăților
- 4. Modificăm starea obiectelor și le rescriem în spațiu
- 5. Eliminăm obiecte din spațiu

Clasa Message

A space stores entries. An entry is a collection of typed objects that implements the Entry interface

```
public class Message implements Entry {
    private String content;
    public Message() {
    }
    public String getMessage() {
        return message;
    }
    public void setMessage(String message) {
        this.message = message;
    }
}
```

Clasa HelloWorldClient

```
public class HelloWorldServer {
    public static void main(String[] args) {
        try {
            // cream un obiect
            Message msg = new Message();
            msg.setContent("Hello World");

            // obtinem o referinta la spatiul de obiecte
            JavaSpace space = SpaceAccessor.getSpace();

            // scriem obiectul in spatiu
            space.write(msg, null, Lease.FOREVER);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Clasa HelloWorldClient (1)

```
public class HelloWorldServer {
    public static void main(String[] args) {
        try {
            // obtinem o referinta la spatiul de obiecte
            JavaSpace space = SpaceAccessor.getSpace();
            space.write(msg, null, Lease.FOREVER);

            // facem o cautare asociativa
            Message template = new Message();
            Message result =
                (Message) space.read(template, null, Long.MAX_VALUE);

            // afisam rezultatul
            System.out.println(result.content);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Clasa HelloWorldClient (2)

```
public class HelloWorldServer {
    public static void main(String[] args) {
        try {
            // obtinem o referinta la spatiul de obiecte
            JavaSpace space = SpaceAccessor.getSpace();
            space.write(msg, null, Lease.FOREVER);

            // facem o cautare asociativa
            Message template = new Message();
            Message result =
                (Message) space.take(template, null, Long.MAX_VALUE);
            result.setMessage("Salutari la toata lumea");

            // modificam si rescriem obiectul
            space.write(result, null, Lease.FOREVER);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Cum rulăm exemplul ?

Avem nevoie de:

- un "web server" (HTTP)
- un "RMI activation server"
- un "Jini lookup service", (eventual RMI registry)
- un "Jini transaction manager"
- un "JavaSpaces server"

RMI