rJava : utilisation parallèle et communication entre deux langages informatiques

Julien Troudet

Pourquoi utiliser rJava (ou JRI) ?

* rJava (Appel de Java dans un script R)
  * Utilisation des Interfaces Homme/Machine (IHM) de Java
  * Bibliothèques spécifiques à Java
  * Parallélisation de certaines taches

* JRI (Appel de R dans un programme Java)
  * Ajout de fonctionnalités statistiques dans une application Java
  * Packages spécifiques à R

Installation de rJava

```
install.packages('rJava')
```

Pour JRI il faut aussi ajouter aux variables d'environnement les localisations des fichiers dll de R et rJava

Possibilité d'utiliser RJava Eclipse Plugin pour faciliter l'utilisation de R dans java

(http://www.studytrails.com/RJava-Eclipse-Plugin/)

Appeler Java depuis R

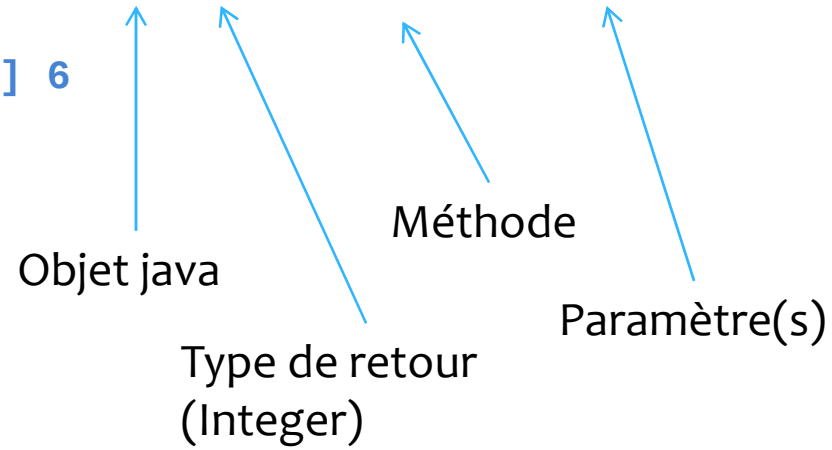
Passage d'instructions basiques à la Java Virtual Machine (JVM)

```
library(rJava)

.jinit()
```
← démarre la JVM

```
s <- .jnew("java/lang/String", "Hello World!")
```
← nouvel objet java (String)

```
.jcall(s,"I","indexOf","World")
```
← appel à la méthode **indexOf** de l'objet **s**

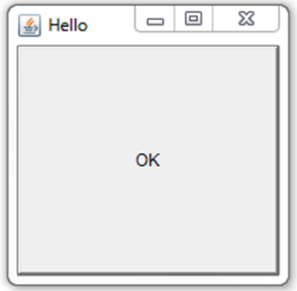
```
[1] 6
```



Appeler Java depuis R

Affichage d'une fenêtre utilisant les composants awt de java (IHM)

```
f <- .jnew("java/awt/Frame", "Hello")

b <- .jnew("java/awt/Button", "OK")

.jcall(f, "Ljava/awt/Component;", "add",
       .jcast(b, "java/awt/Component"))

.jcall(f,, "pack")
.jcall(f,, "setVisible", TRUE)
.jcall(f,, "setSize", as.integer(200), as.integer(200))
```



Créer un package R utilisant Java

```
helloJavaWorld
  - inst
    - doc
      - helloJavaWorld.Rnw
    - java
      - hellojavaworld.jar
  - java
    - HelloJavaWorld.java
  - man
    - helloJavaWorld.Rd
  - R
    - helloJavaWorld.R
    - onLoad.R
  - DESCRIPTION
  - NAMESPACE
```

```
public class HelloJavaWorld {
   public String sayHello() {
      String result = new String("Hello Java World!");
      return result;
   }
   public static void main(String[] args) { }
}
```

```
helloJavaWorld <- function(){
    hjw <- .jnew("HelloJavaWorld")
    out <- .jcall(hjw, "S", "sayHello")
    return(out)
}
```

```
.onLoad <- function(libname, pkgname) {
     .jpackage(pkgname, lib.loc = libname)
}
```

Appeler R depuis Java

Passage d'instructions basiques à R

```
Rengine rengine = new Rengine(new String[]{"--vanilla"}, false, null);
```

nouvelle instance de R

```
int y = 15;
```

← un entier Java

```
rengine.assign("y", y);
```

← Assignation de y à une variable du même nom dans R

```
rengine.eval("yArray <- c(1:reponse)");
```

← Instruction passée à R

```
double[] yArray = rengine.eval("yArray").asDoubleArray();
```

← Récupération de la valeur d'une variable R

```
System.out.println(yArray);
```

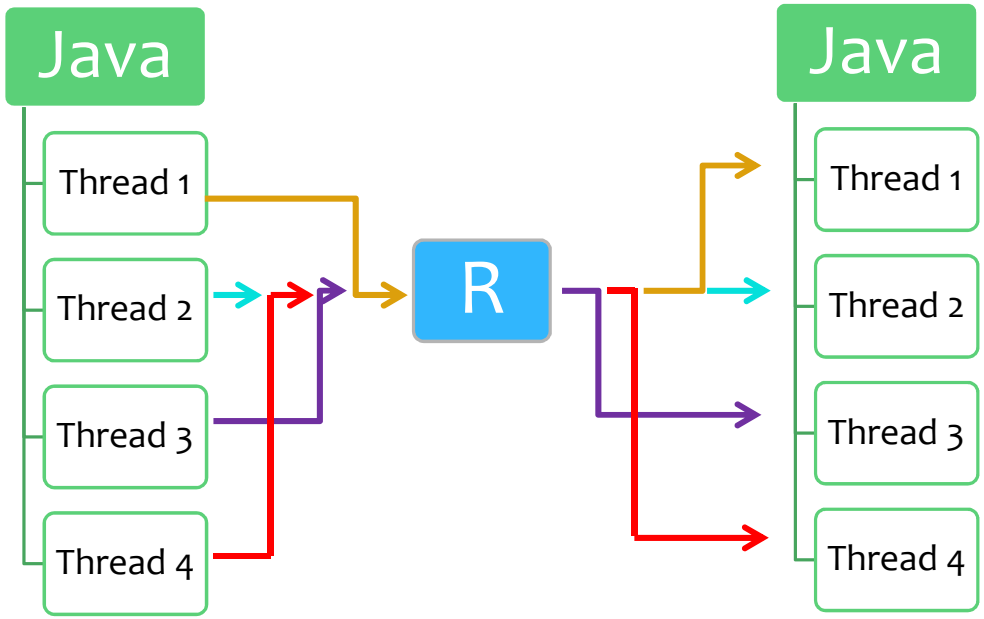
```
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
```

Appeler R depuis Java

Utilisation d'un fichier de scripts

```
Rengine rengine = new Rengine(new String[]{"--vanilla"}, false, null);

//création d'un objet fichier qui pointe vers un script R
File scriptFile = new File("ressources"+File.separator+"script.R");

//modification du chemin (Windows)
String pathForR = scriptFile.getAbsolutePath().replace("\\", "\\\\")

int y = 15;
rengine.assign("y", y);
rengine.eval(String.format("source('%s')", pathForR));
double[] yArray = rengine.eval("yArray").asDoubleArray();
```

Multithreading



R n'est pas conçu pour utiliser plusieurs processeurs, et implémenter cette fonctionnalité peut être compliqué ou impossible selon les cas.

Java est facile à paralléliser et est capable de faire interagir de multiples threads avec une instance unique de R.

Multithreading

```
public void searchForOutliers() {
   int nThreads = Runtime.getRuntime().availableProcessors();
   ExecutorService executorService =
   new ThreadPoolExecutor(
      nThreads, // nombre de thread minimal
      nThreads*2, // nombre max de threads
      1,
      TimeUnit.MINUTES,
      new ArrayBlockingQueue<Runnable>(nThreads, true),
      new ThreadPoolExecutor.CallerRunsPolicy()
   );
   for(Species species : listSp){//boucle sur la liste d'espèces
      executorService.execute(new ProcessusGeoOutliers(species));
   }
    executorService.shutdown();
   try {
       executorService.awaitTermination(3600, TimeUnit.SECONDS);
   } catch (InterruptedException e) {
      e.printStackTrace();
   }
}
```

Création du pool de threads à utiliser et boucle du processus pour détecter les outliers géographiques de chaque espèce.

Multithreading

```
private Species species;
private int xEckert, yEckert;

   public ProcessusGeoOutliers(Species species) {
      super();
      this.species = species;
      this.xEckert = getXEckert(species);
      this.yEckert = getYEckert(species);
   }

   @Override
   public void run() {

      double[] outliers = RScriptExecutor.getGeographicOutliers(xEckert, yEckert);

      this.species.setOutliers  = outliers;
   }
}
```

Le processus lance automatiquement la méthode run() à chaque nouvelle instance.

Cette méthode fait appel à une fonction utilisant R

Multithreading

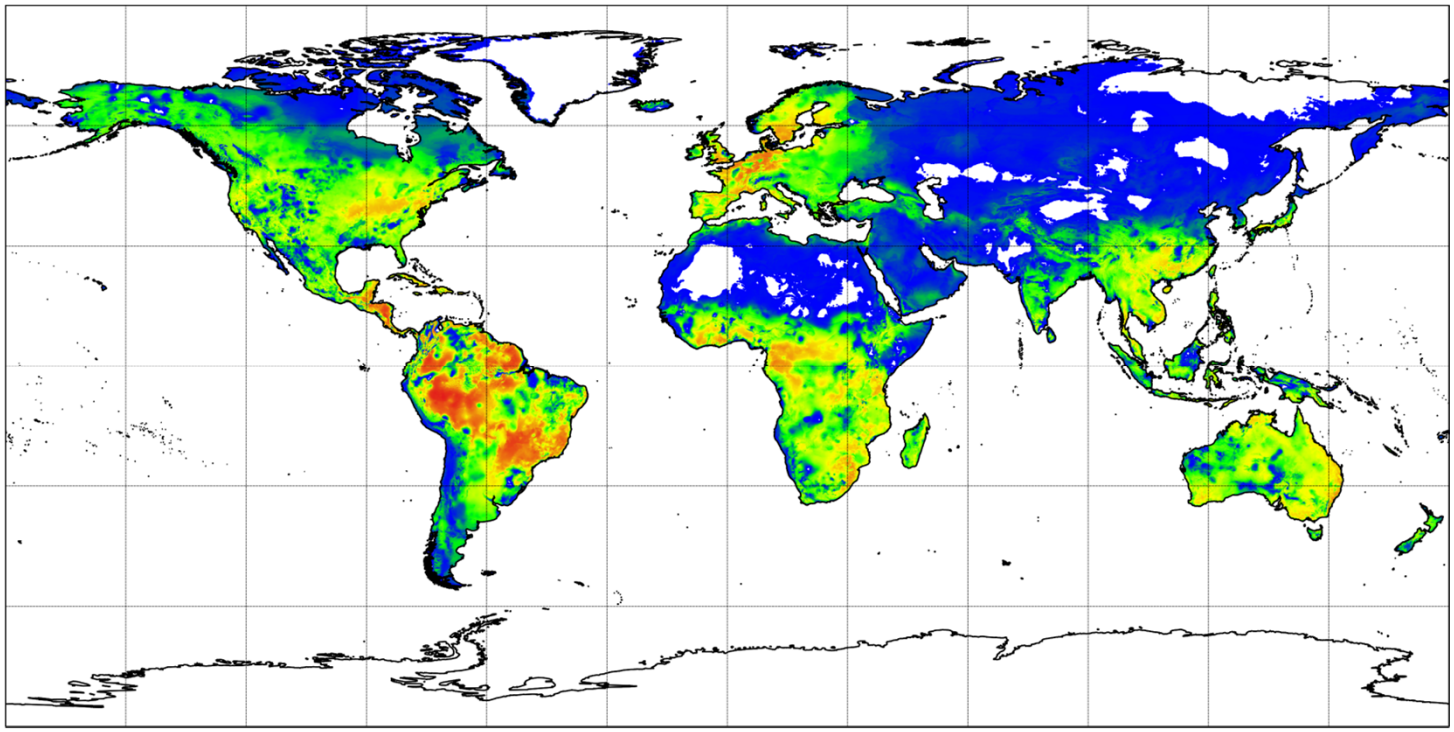
```
public synchronized static double[] getGeographicOutliers(double[] xEckert, double[] yEckert)
{

rengine.assign("xEckert", xEckert);
rengine.assign("yEckert", yEckert);

//read the R file
rengine.eval(String.format("source('%s')", scriptFilePath);

double[] isOutlier  = rengine.eval("outlier").asDoubleArray();

rengine.eval("rm(list=ls(all=TRUE))");

return isOutlier;

}
```

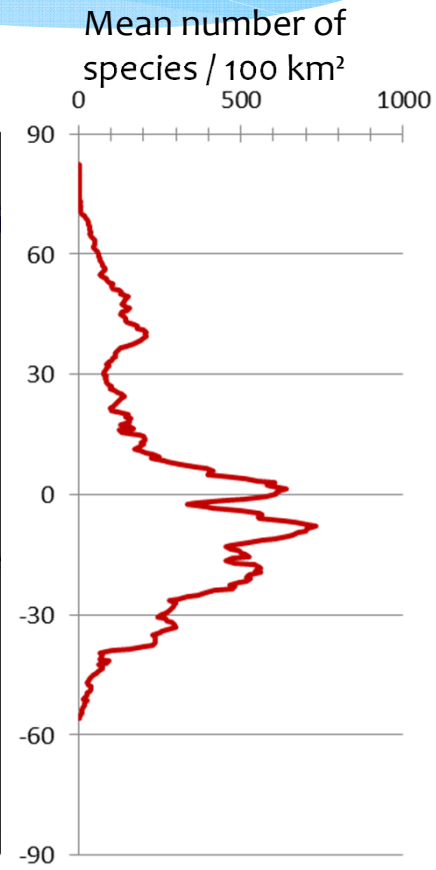
La fonction utilise le mot clé « synchronized » ce qui interdit l'utilisation simultanée de cette méthode dans plusieurs threads.

Traitement de 650 millions d'occurrences

Données après traitement:

- 12 classes
- 33 millions d'occurrences distinctes spatialement
- Plus de 100 000 espèces





Aller plus loin

* Hello Java World! A Tutorial for Interfacing to Java Archives inside R Packages. Tobias Verbeke 2014-09-03

* Rserve → Serveur TCP/IP qui permet de faire interagir des programmes avec un « serveur » R.