Biocep-R

Open Science in the cloud, towards a universal platform for mathematical and statistical computing

Karim Chine
karim.chine@m4x.org
Croire possible le souhaitable est aussi dangereux que de croire souhaitable le possible. Utopies sentimentales et automatismes de la technique.

Nicolás Gómez Dávila

Il n’y a que le solitaire qui soit capable de penser plus que des vérités tactiques.

Nicolás Gómez Dávila
Extract from the GridSolve Description Document

The emergence of Grid computing as the prototype of a next generation cyberinfrastructure for science has excited high expectations for its potential as an accelerator of discovery, but it has also raised questions about whether and how the broad population of research professionals, who must be the foundation of such productivity, can be motivated to adopt this new and more complex way of working.

The rise of the new era of scientific modeling and simulation has, after all, been precipitous, and many science and engineering professionals have only recently become comfortable with the relatively simple world of the uniprocessor workstations and desktop scientific computing tools. In that world, software packages such as Matlab and Mathematica represent general-purpose scientific computing environments (SCEs) that enable users — totaling more than a million worldwide — to solve a wide variety of problems through flexible user interfaces that can model in a natural way the mathematical aspects of many different problem domains.

Moreover, the ongoing, exponential increase in the computing resources supplied by the typical workstation makes these SCEs more and more powerful, and thereby tends to reduce the need for the kind of resource sharing that represents a major strength of Grid computing [1]. Certainly there are various forces now urging collaboration across disciplines and distances, and the burgeoning Grid community, which aims to facilitate such collaboration, has made significant progress in mitigating the well-known complexities of building, operating, and using distributed computing environments. But it is unrealistic to expect the transition of research professionals to the Grid to be anything but halting and slow if it means abandoning the SCEs that they rightfully view as a major source of their productivity. We therefore believe that Grid computing’s prospects for success will tend to rise and fall according to its ability to interface smoothly with the general purpose SCEs that are likely to continue to dominate the toolbox of its targeted user base.
Biocep

R as a Java Toolkit
- Standard R object types mapping
- S4 classes
- Mapping Generator
- Generic Functions
- Annotated Functions
- Distributed Ref Objects
- R Object Names
- R Functions as Java Functions
- Extended Autoboxing
- R Services API

Resource Pooling Framework
- R Objects as HTML
- R Graphics as SVG, JPG, PDF

R as a Web Toolkit
- On-demand Exposure of R functions
- Mapping Generator
- Stateless Web Services
- Stateful Web Services
- R-SOAP API
- Workflows
- JAX-WS

Web Services Generator for R
- Computational Components
- Computational Resources
- Computational Data Storage
- Computational GUIs
- Computational Scripts

The Computational Open Platform
- Diagrams

R as a Server
- Scripting with R
- RMI
- R-HTTP
- R-SOAP
- R-CORBA
- Python
- Groovy
- Java

Parallel & Distributed Computing
- R Ecosystem Exposure
- Scripting Engines APIs
- Nodes / Servers Supervisor
- Dynamic Content generation on-the-fly

R as a Web Toolkit

R Virtualization
- Deployment
- Connection Dialog
- Visual Builder
- Repository
- Built-in Views
- Java Web Start
- Desktop

Virtual R Workbench
- Plugins
- applet
- modes
- R Help
- Academic Grids
- EC2
- Local Machine
- Internet Host

Collaborative R
- Usability
- Deliverables
Biocep Computational Open Platform Ecosystem

### Computational Components
R packages: CRAN, Bioconductor...
Wrapped C, C++, Fortran...
Code
open source or commercial

### Computational Resources
R engines local or remote
intranet machines, clusters, grids, cloud servers
free: academic grids, NGS...
or pay-per-use: EC2, brokers...

### Computational GUIs
Virtual workbench within the browser
Built-in views / Plugins
Collaborative views
Open source or commercial

### Computational Data Storage
Local, NFS, FTP, Storage Web Services (S3)
free or commercial

### Computational Scripts
R / Python / Groovy
On client side: interactivity...
On server side: data transfer...

### Generated Computational Web Services
Stateful or stateless, automatic mapping of R data objects and functions

### Computational Engine API
R as a stateful Web Service
R Virtualization

Server Side - Personal Machine, Academic Grids, Clusters, Clouds

Client Side - Internet

Virtual R Workbench

- Internet Browser
- Java Applet
- Virtual R Workbench URL
- Docking Framework
- R Console
- R Graphic Device+Interactors
- R Workspace
- R Help Browser
- R Script Editor
- R Spreadsheet
- Groovy / Jython Script Editor
Server-side, grid-enabled, collaborative spreadsheet

Cells access from R Console / R scripts
- R functions: 1. `cells.get`  2. `cells.put`  3. `cells.select`

Dynamically evaluated cells using R functions
Paste R expressions into cells
Export cells to R variables

Macros
- User-defined actions: run user R / Groovy / Python scripts
- Events-driven macros 1: on cells change  2: on R variables change
- Data Links: dock R variables in the Spreadsheet: synchronized changes

Collaboration
- Simultaneous viewing of the same data by Jean, Pierre & Paul
- Collaborative cells editing
- Broadcasted cells Selection
Integrating R - State of the art

- **SJJava and rJava/JRI**
  - Basic mapping via JNI of the R C API
- **TypeInfo**
  - Plug meta descriptions to R functions
- **RWebservices**
  - Generated Java Beans for basic R Types / S4 Classes
  - Axis Web Services based on SJJava and ActiveMQ
- **JavaGD**
  - R devices connection to Java (JGR)
- **Rserve**
  - TCP/IP interface to R
What was missing?

- High Level Java API for Accessing R
- Stateful, Resuable, Remotable R Components
- Scalable, Distributed, R Based Infrastructure
- Safe multiple clients framework for components usage as a pool of indistinguishable Remote Resources
- User friendly Interface for the remote resources creation, tracking and debugging
What was missing?

- Generated light-weight Java proxies for R Types / S4 Classes
- On-demand mapping and deployment of R packages as RMI Components or as JAX-WS Web Services
- Remotable R Graphics / Swing Components for R
- Remote R components files exchange API
- Semi-thick client (applet) for web based tools using R
Standard R objects mapping to Java
Generated beans for ExpressionSet
public interface RServices extends ManagedServant {

public String consoleSubmit(String expression) throws ...
public String evaluate(String expression) throws ...

public RObject getObject(String expression) throws ...
public Object getObjectConverted(String expression) throws ...
public RObject getReference(String expression) throws ...
public RObject getObjectName(String expression) throws ...

public void putAndAssign(Object obj, String name) throws ...
public RObject putAndGetReference(Object obj) throws RemoteException;

public RObject call(String methodName, Object... args) throws ...
public RObject callAndConvert(String methodName, Object... args) throws ...
public RObject callAndGetReference(String methodName, Object... args) throws ...
public RObject callAndGetObjectName(String methodName, Object... args) throws ...
public void callAndAssign(String varName, String methodName, Object... args) throws ...

public RObject realizeObjectName(RObject objectName) throws ...
public Object realizeObjectNameConverted(RObject objectName) throws ...
public RObject referenceToObject(RObject refObj) throws ...

public boolean isReference(RObject obj) throws ...
public void assignReference(String name, RObject refObj) throws ...

}
public interface RServices extends ManagedServant {

    public String[] listPackages() throws ...
    public RPackage getPackage(String packageName) throws ...

    public GDDevice newDevice(int w, int h) throws ...
    public GDDevice[] listDevices() throws ...

    public interface GDDevice extends Remote {
        public Vector<GDObject> popAllGraphicObjects() throws ...
        public void fireSizeChangedEvent(int w, int h) throws ...
        public void dispose() throws ...
    }

    public String[] getWorkingDirectoryFileNames() throws ...
    public FileDescription getWorkingDirectoryFileDescription(String fileName) throws ...
    public void createWorkingDirectoryFile(String fileName) throws ...
    public void removeWorkingDirectoryFile(String fileName) throws ...
    public byte[] readWorkingDirectoryFileBlock(String name, long off, int size) throws ...
    public void appendBlockToWorkingDirectoryFile(String name, byte[] block) throws ...

    public String getRHelpFileUri(String topic, String pack) throws ...
    public byte[] getRHelpFile(String uri) throws ...

    public Vector<RAction> popRActions() throws ...
}

RServices API - II
public interface RServices extends ManagedServant {

  public void startHttpServer(int port) throws ...
  public void stopHttpServer() throws ...

  public String pythonExec(String pythonCommand) throws ...
  public RObject pythonEval(String pythonCommand) throws ...
  public void pythonSet(String name, Object Value) throws ...

  public String groovyExec(String groovyCommand) throws ...
  public Object groovyEval(String expression) throws ...
  public void groovySet(String name, Object Value) throws ...

  public void setCallBack(RCallback callback) throws ...

  public String getStatus() throws ...
  public void stop() throws ...
  public void freeReference(RObject refObj) throws ...
  public void freeAllReferences() throws ...
  public String print(String expression) throws ...
  public String sourceFromResource(String resource) throws ...
  public String sourceFromBuffer(StringBuffer buffer) throws ...
  public RNI getRNI() throws ...

  ...
}

RServices API - III
Remote Resources Pooling Framework

• Generic Standalone framework
• Pooling of any RMI components and if combined with JNI of any library / open architecture

• New Remote Object Registry based on Derby| Oracle| MySQL

• Three implementations available
  - rmiregistry / mono-node / single client process
  - rmiregistry / multinodes / single client process
  - database ROR / multinodes / multiple client processes

• User friendly interface for the remote resources creation, tracking and debugging, nodes and pools management
Computational Engines Pools

Remote Objects Registry

Node 1: Windows XP
- Front-end host
- R-HTTP
- R-SOAP
- .NET Appl
  - logOn
  - Use R
  - logOff
- Perl Scripts
  - logOn
  - Use R
  - logOff
- Web Application
  - Borrow R
  - Use Rs
  - Release Rs

Node 2: Mac OS
- Computational Engines Pools
  - Pool A
  - Pool B
  - Pool C
- Supervisor
- Cloud bursting via Amazon Web Services

Node 3: 64 bits Server / Linux
- Parallel Computing Applications
  - Borrow Rs
  - Use Rs
  - Release Rs

Node 4: EC2 virtual machine 1

Node 5: EC2 virtual machine 2
R Pools

Supervisor

Object Export / Import Layer

RServices API

Client Application

JVM

rJava / JRI

JavaGD

mapping.jar

Remote Objects Registry

RServices skelton

R packages skeltons

R graphic device skelton

Pooling framework

Browser( java plugins( applet ) )

Tomcat

Pooling framework

tunneling servlet

Generated mapping

JAX-WS servlet/artifacts

Pooling framework

Tomcat

Http Tunneling

SOAP

.NET, Perl.. Application
Scripting

File System

Object Export / Import Layer

RServices API

RServices skeleton R graphic device skel R packages skels

JVM

rJava / JRI JavaGD

mapping.jar

Client

Server

Virtual R Workbench

Open Swing input Dialog

Client Side Groovy Script

import javax.swing.JOptionPane;
n=JOptionPane.showInputDialog(null, 100);
n=Integer.decode(n); client.R.getInstance().putAndAssign(n,"n")

if (n%2==0) {
  <R>
  hist(rnorm(n))
  </R>
}

else {
  <R>
  plot(rnorm(n))
  </R>
}
final double[][] m=..;
Future<Double>[] result=new Future[m.length];
ExecutorService exec = Executors.newFixedThreadPool(50);
for (int i=0; i<result.length; ++i) {
    final double[] v=m[i];
    result[i]= exec.submit(
        new Callable<Double>() {
            public Double call() throws Exception {
                RServices r=null;
                try {
                    r=(RServices)ServantProviderFactory .getFactory().ge tServantProvider().
                    borrowServantProxy();
                    Rnumeric mean=(RNumeric)r.call("mean", new RNumeric(v));
                    return mean.getValue()[0];
                } finally {    ServantProviderFactory .getFactory().getServantProvider().
                    returnServantProxy(r)    
                }
            }
        });
}
while(true) {
    int count=0; for (int i=0; i<result.length; ++i) if (result[i].isDone()) ++count; if (count==result.length) break;
    Thread.sleep(100);
}
for (int i=0; i<result.length; ++i) System.out.println(result[i].get());
**Snow with Biocep**

*From the R Console:*

- `makeCluster(n,...)`  `stopCluster(cl)`  
  - Starting and Stopping clusters

- `clusterEvalQ(cl, expr)`
  - The expression is evaluated on the slave nodes.

- `clusterApply(cl, seq, fun, ...)`
  - Calls the function with the first element of the list on the first node, with the second element of the list on the second node, and so on.

- `clusterExport(cl, list)`
  - Assigns the global values on the master of the variables named in 'list' to variables of the same names in the global environments of each node.

...
Web Services Generation

**Script / globals.r**

```r
square <- function(x) {return(x^2) }
typelnfo(square) <- SimultaneousTypeSpecification(
  TypedSignature(x = "numeric"), returnType = "numeric")
```

**Script / rjmap.xml**

```xml
<rj>
  <functions>
    <function name="square" forWeb="true"/>
  </functions>
  <scripts>
    <initScript name="globals.r" embed="true"/>
  </scripts>
</rj>
```

**WS generator**

- Mapping jar
- Pooling framework
- R Java Bridge
- JAX-WS
  - Servlets
  - Generated artifacts

**rws.war**

- WS generator
- Deploy
- R HTTP
- Tomcat

**WSDL**

http://127.0.0.1:8080/rws/rGlobalEnvFunction?WSDL

**Eclipse Web Service Client Generator**

```java
public static void main(String[] args) throws Exception {
  RGlobalEnvFunctionWeb g=new RGlobalEnvFunctionWebServiceLocator().getRGlobalEnvFunctionWebPort();
  RNumeric x=new RNumeric(); x.setValue(new Double[]{6.0});
  System.out.println(g.square(x).getValue()[0]);
}
```
Workflows with Stateful Web Services

SessionID associated with a reserved R worker

LogOn, getData : R-SOAP methods
ES : ExpressionSet
ESon1, ESon2, ESon3 : ExpressionSet Object Names
f = T3 o T2 o T1
R Virtualization on an LSF Cluster

LSF Submission Host

LSF Node 1

LSF Node 2

LSF Node 3

Shared File System 1

Shared File System 2

DMZ

INTERNET

RMI

SSH

SSH

Front-end Host

R Servers Manager

Sessions Manager

Tunneling Servlet

Tomcat

biocep-core Generated mapping

JAX-WS servlet/artifacts

Http Tunneling

Java Applications

HTTP Tunneling

Java,.NET,perl Applications

R Virtualization

Tunneling Sessions

Servlet Manager

Java,.NET,perl Applications

Http tunneling

 ↔ Serialized Java Objects

Virtual R Workbench

Http tunneling

 ↔ Serialized Java Objects

create process

kill process

http tunneling

RMI

SSH

SSH

Front-end Host

R Servers Manager

Sessions Manager

Tunneling Servlet

Tomcat

biocep-core Generated mapping

JAX-WS servlet/artifacts

Http Tunneling

Java Applications

HTTP Tunneling

Java,.NET,perl Applications

R Virtualization

Tunneling Sessions

Servlet Manager

Java,.NET,perl Applications

Http tunneling

 ↔ Serialized Java Objects

Virtual R Workbench

Http tunneling

 ↔ Serialized Java Objects

create process

kill process

http tunneling
Netbeans 6 – Visual GUI builder

GUI Plugins

myPlugin.jar
+ myView1
+ myView2
+ descriptor.xml

Compile

Import Plugin

Virtual R Workbench

Upload plugin

Plugins Repository
* myPlugin  * myDashboard
* Klimt  * iPlots
* Mondrian  * E. Profiler

Browse Repository

Download Plugin
Collaborative R

 INTERNET

 DMZ

 Same R session for U1, U2 & U3
 Broadcasted console + chat

 Broadcasted Main R Graphic Device

 Collaborative Script editor

 Collaborative Spreadsheet

 Same virtual workspace for U1, U2 & U3

 Collabortive R

 INTERNET

 DMZ

 INTERNET

 DMZ

 ACADEMIC GRIDS, NGS, EC2, INTRANET LSF, INTRANET HOST..
Ease Of Use - I

• Reasonable Pre-requisites
  • Java 5 and R>=2.5 accessible from the command line: to run R servers, generate
  • mappings & Web Services, run the miniature virtualisation and the R-SOAP Web Apps.

• All-in-one Highly Productive Workbench
  • Docking framework, spreadsheets, syntax highlighting enabled editors, objects viewer,
    • help browser, storage views, zooming system on R graphics, settings persistence.

• Easy Computational Resource Acquisition
  • Provide nothing to run R servers on local machine
  • Provide HOST / PORT / LOGIN / PWD to run R Servers on remote hosts (SSH)
  • Provide URL & (LOGIN/PWD or X.509 Certificate) to Connect to Grid Rs or Cluster Rs

• Easy Scripting
  • Simple API for running/connecting to R servers
  • Embeddable R code (<R> </R>) within scripts
  • Automatic conversion from/to R Objects for common data types(standard, arrays, collections)
• **Easy Plugins Integration**
  Import local file / Browse Plugins repository and choose a plugin

• **« Push button » Web Services Generation/Web Services Deployment**
  Add TypeInfo to your function / add your function name to an XML / run biocep-tools
  Deploy: `java -port=80 -cp biocep-core.jar HttpServer rvirtual.war MyWebServices.war`

• **Self-contained jar & war files distribution :**
  biocep.jar biocep-core.jar biocep-tools.jar rvirtual.war rws.war

• **Configurationless Parallel Computing from R console :**
  makeCluster(n,..), stopCluster(cl), clusterEvalQ(cl, expr), clusterApply(cl, seq, fun, ..) ...
Acknowledgements

www.biocep.net