Scientific Computing Environments in the age of virtualization

Toward a universal platform for the Cloud

Karim Chine
Cloud Era Ltd
Cambridge, UK
karim.chine@polytechnique.org
www.biocep.net

Abstract— **This paper describes Biocep-R, an Open Source platform for the virtualization of Scientific Computing Environments (SCEs) such as R and Scilab. To our knowledge it is the first time that a software platform enables geographically distributed collaborators to view and analyze terabytes of data interactively and collaboratively, using standard computational tools. Those tools can be running on high performance machines or on a Cloud. This is also the first time that a full end-to-end solution is proposed for reproducible computational research in a Cloud and for virtual appliances-based education.**

Keywords: *HPC; cloud computing; distributed computing; application virtualization; SaaS; Web Services; workflows; cyberinfrastructure; large scale data mining; collaborative data analysis; reproducible research; open source*

I. Introduction

R is a language and environment for statistical computing and graphics. It is becoming the *lingua franca* of data analysis. Repositories of contributed R packages related to a variety of problem domains in life sciences, social sciences, finance, econometrics, chemometrics, etc. are growing at an exponential rate. Scilab is a scientific software package for numerical computations providing a powerful open computing environment for engineering and scientific applications. Biocep-R is a GPL Java platform that enables to use R, Scilab or any other computational environment with an API (example: Sage, Octave, Root, Matlab, SAS, etc.) interactively on Clusters, Grids or private/public Clouds and to enable the interoperability, pluggability, sharing and reuse of the computing artifacts.

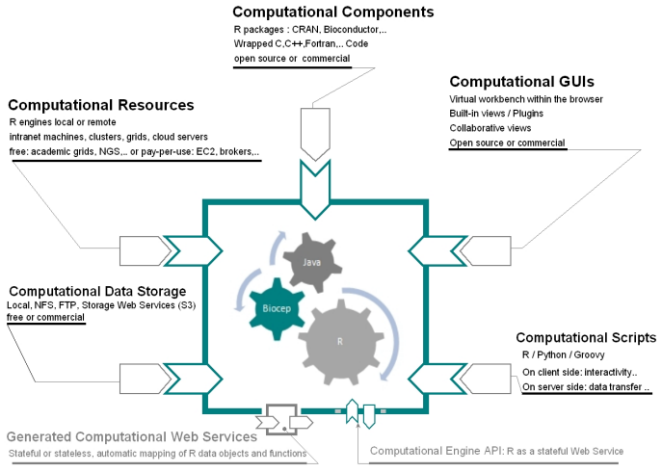

FIG. 1 Biocep-R Computational Open Platform

II. Lowering The Barriers for accessing cyber infrastructures. Local/Remote transparency

The same application (the virtual workbench) makes it easy to connect to various environments locally or on remote machines whether they are nodes of a grid or virtual machines of a Cloud. Using Java remoting technologies and HTTP relays, Biocep-R makes it possible to uniquely identify a remote generic computational engine with a simple URL: Switching from one resource to another (EGEE to TeraGrid to Amazon's Cloud) becomes as simple as replacing one URL with another.

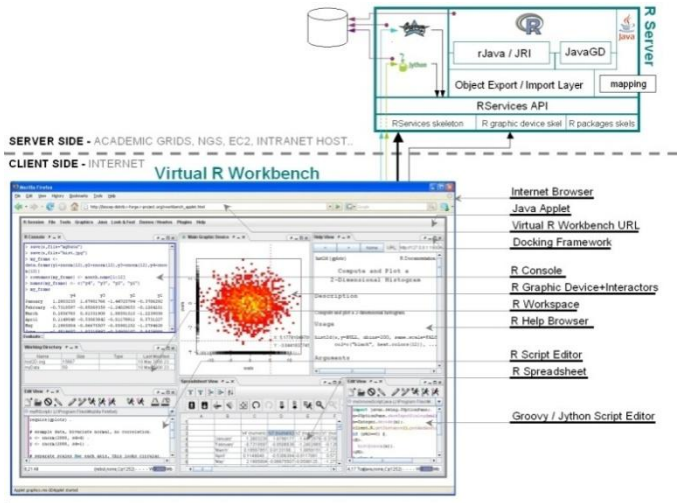

FIG. 2 R Virtualization

The virtual workbench has several dockable built-in views including:

- Consoles for issuing commands to the SCEs or to the server-side scripting interpreters (Python, Groovy, Ruby, *etc.*)
- Remote working directory browser.
- Syntax-highlighting-enabled code editors.
- Help viewers.
- Viewers for PDF, SVG, HTML, *etc.* files.
- Highly interactive server-side graphic devices (with built-in zooming, scrolling, coordinate tracking, *etc.*).
- Data inspectors.
- Server-side linked plots.
- Server-side spreadsheets that are fully integrated with the SCEs functions and data, *etc.*

Biocep-R is currently available on Amazon's Elastic Cloud. Here are the steps a user needs to follow to perform computing on EC2 (see www.biocep.net for details):

- Sign up for EC2
- Use Amazon's Elasticfox Graphic User Interface (GUI) to browse the Biocep-R AMIs (Amazon Machine Images) and choose the one corresponding to the computing environment and libraries he needs.
- Choose an instance type (memory size, number of cores…), provide his email in the user data and run the AMI.
- When the AMI starts running (~2 minutes wait time) he receives an email containing a URL
- The user can then
    - Click on the URL and this runs the Biocep-R virtual workbench which connects automatically to a computational engine on the running machine instance. drag-and-drops his R scripts and his data files from his desktop to the virtual working directory view
    - Execute R scripts using the R console
    - Drag-and-drop result files to his desktop
    - Shut down the AMI when the session is no longer needed.

Besides the virtual workbench, the RESTful Biocep-R server enables users to compute and generate graphics on HPC resources using only a browser. Simple URLs allow them to execute any script or to evaluate any expression by workers from a back-end computational engines pool and to retrieve the results either as text files or as Graphics in any format (pdf, svg, jpeg, png, etc).

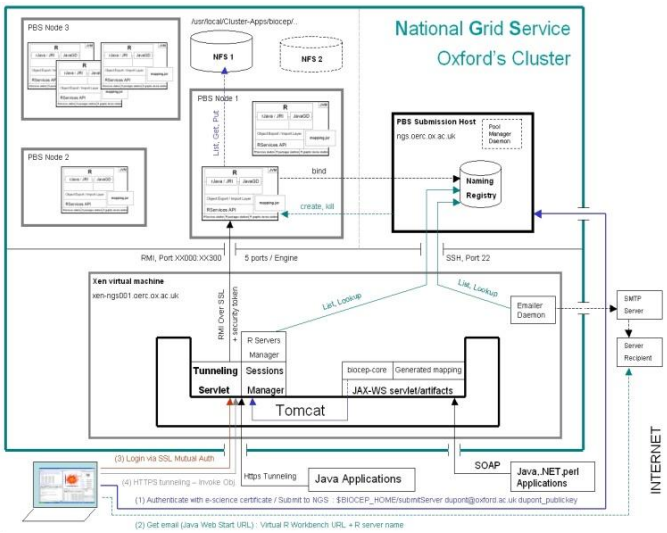

FIG. 3 R virtualization on the National Grid Service

III. DEALING WITH THE DATA DELUGE

The data generated by modern science tools can become too large to move easily from one machine to another. This can be an issue for large collaborative projects. The analysis of such data can't be performed the way it has been so far. The answer to this increasingly acute problem is to take the computation to the data and is what Biocep-R enables cyberinfrastructures' users to do: The generic computational engine can run on any machine that has a privileged connectivity with the data storage machine or within the large scale database. The user can connect his Biocep-R virtual workbench (or his scripts using the Biocep-R SOAP clients) to the computational engine, set the working directory to the location of the data (e.g. via NFS) and view or analyze the data using R/Scilab packages.

IV. ENABLING COLLABORATION WITHIN COMPUTING ENVIRONMENTS

Users can connect to the same remote engine and work with large scale data collaboratively using broadcasted commands/graphics and collaborative spreadsheets. For example, the Amazon EC2 user can forward the email received from the Biocep-R running AMI to any number of his collaborators. By clicking on the same URL, they all get connected to the same computing environment. Every command issued by one of them is seen by all the others. Synchronized R graphics panels allow them to see the same graphics and to annotate them collaboratively. Chatting is enabled. Views based on a refactored iplots package enable collaborative highlighting and color brushing on a variety of high interaction graphics (linked plots).

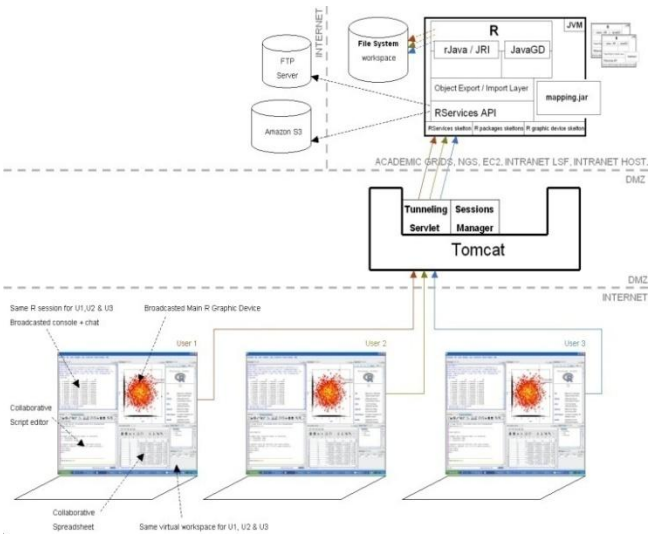

FIG. 4 Collaborative R

V. SCIENCE GATEWAYS MADE EASY

Web-based interfaces and portals allowing scientists to use a Grid to solve their domain specific problems have always been difficult to develop, upgrade and maintain. We should have front ends that are easy to create. Biocep-R proposes a different paradigm for the creation and distribution of such front-ends to HPC/Cloud environments.

1) The Biocep-R Plug-ins

The Biocep-R platform defines a contract for creating cross-platform statistical/numerical new interfaces in Swing-Java either programmatically or using visual composition tools like the Netbeans GUI designer. The views can be bundled

into zip files and opened by anyone using the Biocep-R virtual workbench. The views receive a Java Interface that allows them to use the R/Scilab engine to which the workbench is connected and that can be running at any location.

Three-parts-URLs (Biocep-R's Java Web Start trigger + computational engine's URL parameter + plug-in's zip file URL parameter) can be used to deliver those GUIs to the end-user. He retrieves them in one click and the only software required to be preinstalled on his machine is a Java runtime. Instead of requiring a transparent connection to a server-side Grid/Cloud-enabled engine, the distribution URLs can be written to trig transparently the creation of a computational engine on the user's machine: a zipped version of R is copied on the user's machine (with or without administrative privileges) and is used transparently by the GUI.

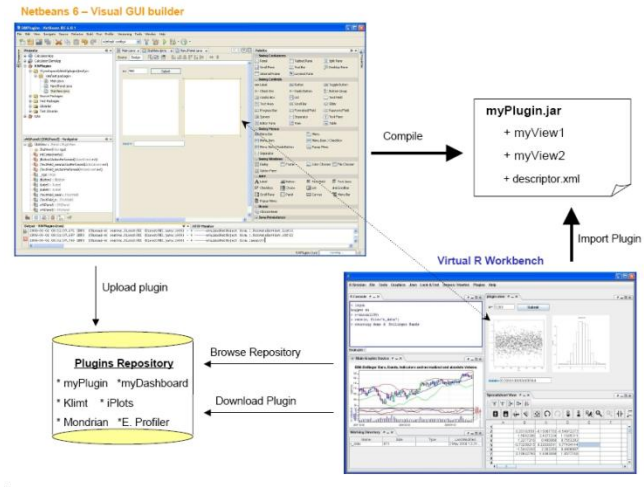

FIG. 5 GUI Plug-ins

2) The Biocep-R Spreadsheets

The Biocep-R spreadsheets are Java-based built originally using the OSS jspreadsheet. Unlike jspreadsheet, Calc and Excel's spreadsheets, they have their models on server-side, are HPC and collaboration enabled and are fully connected with the remote statistical/numerical engine's workspace. This enables for example R data import/export from/to cells and R functions use in formula cells. Dedicated R functions (cells.get, cells.put, cells.select, *etc.*) allow the R user to retrieve the content of cell ranges into the R workspace or to update them programmatically: An R script can reproduce entirely the spreadsheet. A macros system allows the user to define listeners on R variables and on cell ranges and to define corresponding actions as R/Java scripts. Specific macros called datalinks allow the user to bi-directionally mirror R variables with cell ranges. R graphics and User Interface components can be docked onto cell ranges. UI components can be for example:

- sliders mirroring R variables
- Graphic Panels showing R Graphics (in any format) produced by user defined R scripts and automatically updated in case user-defined R variables have their values change or in case cells within a user-defined cell ranges list are updated
- Buttons executing any user-defined R script, *etc.*

This spreadsheet enables scientists without programming skills to create sophisticated Grid/Cloud-based analytical views and dashboards and lowers the barriers for creating science gateways and distributing them.

VI. Bridging the Gap between Existing SCEs and Grids/Clouds

Once the user's workbench is connected to a remote R/Scilab engine, a RESTful embedded server (local http relay) enables third-party applications such as emacs, Open Office Calc or Excel to access and use the Grid/Cloud-enabled engine. For example, an Excel add-in is being built to use the full capabilities of the platform and reproduce the features of the Biocep-R spreadsheets from within Excel. The bi-directional mirroring of server-side spreadsheets' models into Excel cell ranges will also be available. This will allow users to overcome some of the Excel flaws (limited capabilities in statistical analysis, inaccurate numerical calculations at the edge of double, inconsistent identification of missing observations...). Excel becomes a front-end of choice to Grid/Cloud resources and can then become the universal workbench for different sciences.

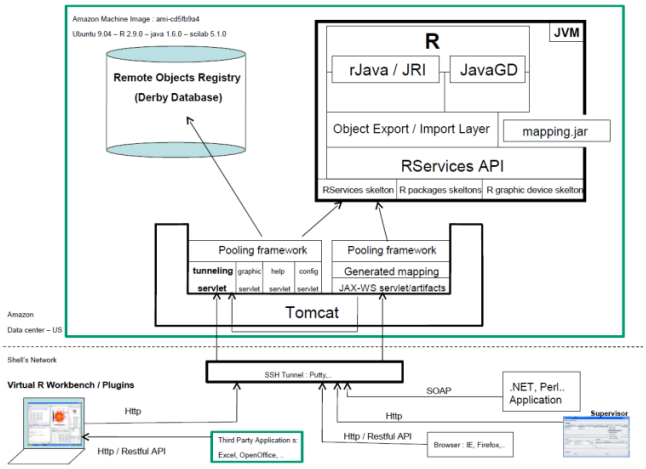

FIG. 6 R On Amazon's Elastic Cloud EC2

VII. A Universal Computing Toolkit for Scientific Applications

Biocep-R frameworks and tools make it possible to use R as a Java object-oriented toolkit or as an RMI server. All the standard R objects have been mapped to Java and user defined R classes can be mapped to Java on demand. R functions can be called from Java as if they were Java functions. The input parameters are provided as Java objects and the result of a function call is retrieved as a Java object. Calls to R functions from Java locally or remotely cope with local and distributed R objects. The full capabilities of the platform are exposed via a SOAP and RESTful front-ends. Several tools and frameworks are provided to help building analytical desktop/web applications and scalable data analysis pipelines in any programming language (Java, C#, C++, Perl, *etc.*)

VIII. SCALABILITY FOR COMPUTATIONAL BACK-ENDS

Biocep-R provides a pooling framework for distributed resources (RPF) allowing pools of computational engines to be deployed on heterogeneous nodes/virtual machines instances. These engines are managed and used via a simple borrow/return API for multithreaded web applications and web services, for distributed and parallel computing, for dynamic content on-the-fly generation (analytic results, tables and graphics in various formats for thin web clients) and for computational engines' virtualization in a shared computational resources context. The engines become agnostic to the hosting operating system. Several tools are provided to monitor and manage the pools programmatically or interactively (Supervisor UI). The pooling framework enables transparent cloudbursting: Amazon EC2 virtual machines instances hosting one or many computational engines can be fired up or shut down to scale up or scale down according to the load in a highly scalable web applications deployment for example.

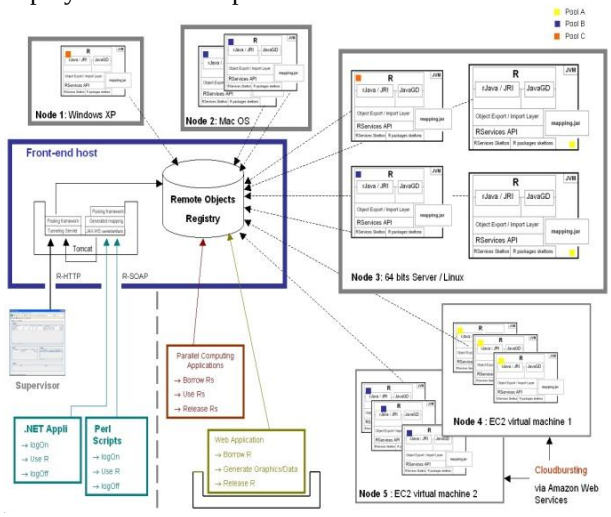

Fig. 7 R engines pools deployment – Cloudbursting

IX. DISTRIBUTED COMPUTING MADE EASY

To solve heavily computational problems, there is a need to use many engines in parallel. Several tools are available but they are difficult to install and beyond the technical skills of most scientists. Biocep-R solves this problem. From within a main R session and without installing any extra toolkits/packages, it becomes possible to create logical links to remote R/Scilab engines either by creating new processes or by connecting to existing ones on Grids/Clouds. Logical links are variables that allow the R/scilab user to interact with the remote engines. *rlink.console*, *rlink.get*, *rlink.put* allow the user to respectively submit R commands to the R/Scilab worker referenced by the rlink, retrieve a variable from the R/scilab worker's workspace into the main R workspace and push a variable from the main R workspace to the worker's workspace. All the functions can be called in synchronous or asynchronous mode. Several rlinks referencing R/Scilab engines running at any locations can be used to create a logical cluster which enables to use several R/Scilab engines in a coordinated way. For example, a function called *cluster.apply* uses the workers belonging to a logical cluster in parallel to apply a function to a large scale R data.

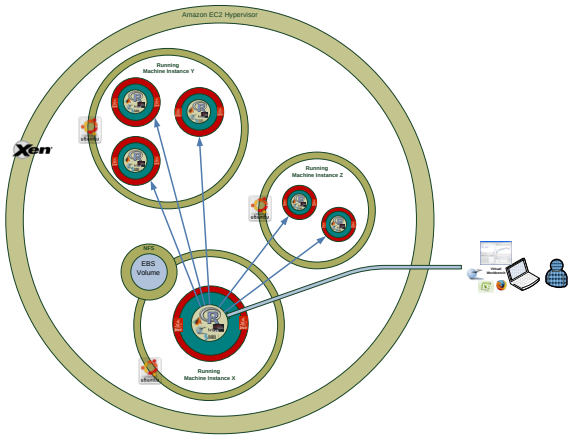

FIG. 8 Distributed Computing on EC2

X. BRIDGING THE GAP BETWEEN MAINSTREAM SCES

The platform has a server-side extensions architecture that enables the creation of bridges between the remote computational engine and any third party tool. Besides R and Scilab, several widely used environments will be integrated in the future (Matlab, Root, SAS, *etc.*). Since R and Scilab are running within the same process (same Java Virtual Machine), it is easy and very fast to exchange data between them. This can be achieved for example by using the Groovy interpreter available as part of the remote engine. The Python client provided by the platform makes it possible for the Scipy community to use R/Scilab engines on Grids/Clouds directly from within their python scripts.

XI. BRIDGING THE GAP BETWEEN MAINSTREAM SCES AND WORKFLOW WORKBENCHES

Biocep-R enables automatic exposure of R functions and packages as Web Services. The generated Web Services are easy to deploy and can use back-end computational engines running at any location. They can be seamlessly integrated as workflows nodes and used within environments such as Knime, Taverna or Pipeline Pilot. They can be stateless (an anonymous R worker performs the computation) or stateful (an R worker reserved and associated with a session ID is used and can be reused until the session is destroyed). The statefulness solves the overhead problem caused by the transfer of intermediate results between workflow nodes.

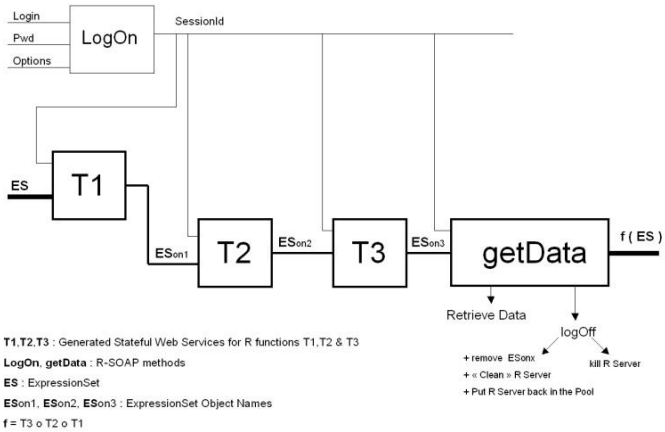

FIG. 9 Generated stateful Web Services workflows

XII. The Building Blocks of a Platform for Statistics and Applied Mathematics Education

Besides being free and open source and therefore accessible to students and educators, Biocep-R provides education-friendly features that only proprietary software could offer so far (for example the centralized and controlled server-side deployment of the Scientific Computing Environments) and enables new scenarios and practices in the teaching of statistics and applied mathematics. With Biocep-R, it becomes possible for educators to hide the complexity of R, Scilab, Matlab, *etc.* with User Interfaces such as Biocep-R plugins/spreadsheets. These are very easy to create and to distribute to students. The User Interfaces reduce the complexity of the learning environment and keep beginning students away from the steep learning curves of R, Scilab or Matlab. Once created by one educator, the User Interfaces can be shared, reused and improved by other educators. Dedicated repositories can be provided to centralize the efforts and contributions of the community of educators and help them sharing the insight gained in using this new environment. One could envisage these methods being used from primary schools to graduate-level studies.

Virtual appliances (VMWare/Virtualbox/Zen virtual machines) prepared by educators can be provided to students on USB keys. The virtual machines contain the SCE, the libraries used for the course and Biocep-R. The students need only to have Java and a virtual machine player (the free VMware player for example) installed on their laptops to run the virtual Biocep-R workbench and to connect to a computational engine on the virtual machine. The virtual appliance is fully self-contained: the code needed to run the workbench or the plug-ins prepared by the educator is delivered by the virtual appliance itself thanks to the Biocep-R code server that runs at startup. The interaction between the student and the SCE as well as the artifacts he produces are saved within the Biocep-R-enabled-virtual machine. The educator can retrieve the USB keys used by the students and checks not only the validity of the different intermediate results they obtained but also the path they followed to get those results.

The collaboration capabilities of the virtual workbench open also new perspectives in distributed learning. The Educator can connect anytime to the SCEs of students at any location. He can then see/update their environments and guide them remotely. Collaborative problem solving becomes also possible and can be used as a support for learning.

XIII. The Building Blocks of a Traceable and Reproducible Computational Research Platform

We provide a system so that the computational environment, the data and the manipulations of the data (scripts, applications) can be recorded. These can be used by reviewers, collaborators and anyone wanting to investigate the data. Biocep-R provides an end-to-end solution for traceable and reproducible computational research. Snapshots of computational environments can be created as virtual machine images (AMIs). Snapshots of versioned libraries and working directories can be created as Elastic Block Stores (EBSs). The Biocep-R virtual workbench makes it possible to all scientists to work with these snapshots (AMIs + EBSs) and produce them easily. By Providing the Biocep-R-enbled AMI identifier, the complementary computational libraries EBS snapshots identifiers and the working directory (data) EBS snapshot identifier that have been used for his research, the scientist makes it possible to anyone to rebuild all the data and the computational environment required to process that data.

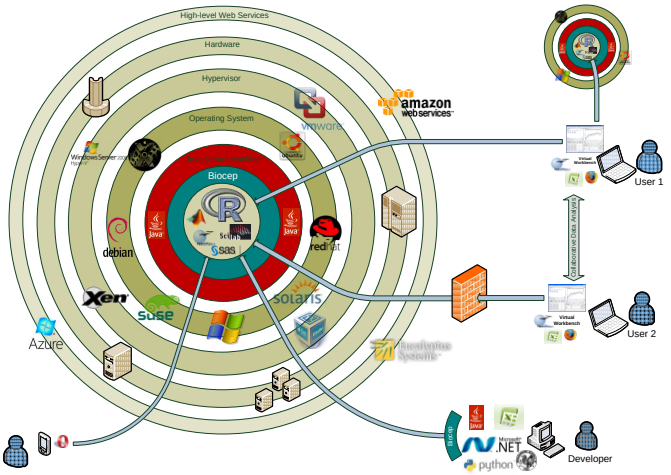

FIG. 10 Biocep-R within the Technology Ecosystem

XIV. Conclusion

This new environment has the potential de democratize the cloud and to push forward the reproducibility of computational research. Its current availability and easy access on amazon's Elastic cloud and its planned deployments on major Grids (NGS, EGEE, TeraGrid) maximize its chances for uptake and adoption. Academia, Industry and Educational Institutions would benefit from the emergence of a new environment for the interoperability, sharing and reuse of computational artifacts. The creation and sharing of analytical tools and resources can become accessible to anyone (open science). An international portal for on demand computing (www.elasticr.net) is being built using the different frameworks provided by Biocep-R and could become a single point of access to Virtualized SCEs on public servers and on virtual appliances that are ready for use on various clouds. There is no question about the need for more usability in the computational landscape. Java, Xen, EC2, R and Biocep-R prove that the target of a universal computational environment for science and for everyone is definitely within reach.

Reference

[1] R Development Core Team (2009). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

[2] Hardt, M., Seymour, K., Dongarra, J., Zapf, M., Ruiter, N.V. "Interactive Grid-Access Using Gridsolve and Giggle," Computing and Informatics, Vol. 27, No. 2, 233-248, ISSN 1335-9150, 2008.

[3] http://www.scilab.org

[4] Theus, M. and Urbanek, S. (2008) Interactive Graphics for Data Analysis: Principles and Examples, CRC Press, ISBN 978-1-5848-8594-8