Chapter 19

Open Science in the Cloud: Towards a Universal Platform for Scientific and Statistical Computing

Karim Chine
Cloud Era Ltd, Cambridge, UK

1. Introduction

The UK, through the e-Science program, the US through the NSF-funded cyber infrastructure and the European Union through the ICT Calls aimed to provide "the technological solution to the problem of efficiently connecting data, computers, and people with the goal of enabling derivation of novel scientific theories and knowledge" [1]. The Grid [2] [3], foreseen as a major accelerator of discovery, didn't meet the expectations it had excited at its beginnings and was not adopted by the broad population of research professionals. The Grid is a good tool for particle physicists and it has allowed them to tackle the tremendous computational challenges inherent to their field. However, as a technology and paradigm for delivering computing on demand, it doesn't work and it can't be fixed. On one hand, "the abstractions that Grids expose – to the end-user, to the deployers and to application developers – are inappropriate and they need to be higher level" [4], and on the other hand, academic Grids are inherently economically unsustainable. They can't compete with a service outsourced to the Industry whose quality and price would be driven by market forces. The virtualization technologies and their corollary, the Infrastructure-as-a-Service (IaaS) style cloud, hold the promise to enable what the Grid failed to deliver: a sustainable environment for computational sciences that would lower the barriers for accessing federated computational resources, software tools and data; enable collaboration and resources sharing and provide the building blocks of a ubiquitous platform for traceable and reproducible computational research. Amazon Elastic Compute Cloud (EC2) [22] is an example of Infrastructure-as-a-Service that anyone can use today. Its considerable success announces the emergence of a new era. However, bringing that era for research and education still requires the software that would bridge the gap between the cloud and the scientists' everyday tools and would make Infrastructure-as-a-Service a trivial commodity.

This article describes Elastic-R [9][10], a software platform that makes working with R on the cloud as simple as working with it locally. More generally, it aims to be the missing link between the cloud and the most widely used data analysis tools and Scientific Computing Environments (SCEs). Elastic-R synergizes the usage scenarios of those environments

with the usage scenarios of the cloud and empowers them with what the cloud has best to offer:

User-friendly and flexible access to Infrastructure-as-a-Service: The cloud interfaces are simple and expose the right abstractions for managing and using virtual appliances within a federated computing environment but the cloud consoles remain tools for computer savvies. Elastic-R offers a simplified façade to the cloud that makes any scientist able to choose and run the virtual machine with the specific scientific computing environment (example: R version 2.9, Scilab [6], Sage [12], Root [13], etc.). The scientist can then have access to the full capabilities of the environment using the Elastic-R Java workbench or from within a standard web browser using the Elastic-R Ajax workbench. The scientist can issue commands, install and use new packages, generate and interact with graphics, upload and process files, download results, create and edit R-enabled server-side spreadsheets, etc. The scientist can disconnect from the engine and reconnect again from anywhere, retrieving his full session including workspace, graphics, etc. and can continue working from where he left off. The virtual machine can be simply shut down when not needed anymore. The user is charged only for the usage time. User's data (working directory content) remains on a virtual disk on the cloud that can be attached once again to a new virtual machine instance.

Collaboration: A virtual machine instance on the cloud has a public IP address and can be seen and used by the owner's collaborators who can be located anywhere. Elastic-R allows the scientist to expose his machine and his R sessions (for example) to his collaborators. All of them can connect their Java or Ajax workbenches to the same R engine and can control that engine and update its environment. The actions of each collaborator in the console (commands issued, chat) , on the graphics (plotting, annotating, resizing and slides viewing) or on the spreadsheets (cells updating, cells selecting) are broadcasted to the others and their workbenches show the changes in real time.

On-demand elasticity: Elastic-R exposes to the scientist a major feature of the cloud which is the ability to choose the capacity of the virtual machine instances, such as the number of virtual cores, the memory size and the disk space. He can then run on the cloud analysis or simulations that require more memory than available on his laptop or would take days if run locally. Elastic-R allows the scientist to solve compute-intense problem by starting any number of virtual machines hosting R engines that can process in parallel partial tasks. Those pools of engines can also be used to

create Web applications with dynamic analytical content generated by R or any other environment. For those applications, the Elastic-R platform enables cloudbursting: virtual machines can be fired up or shut down (increasing or decreasing the engines pool size) to scale up or scale down according to the load of the application.

Applications deployment flexibility: The cloud can host very easily client-server style applications. Elastic-R is a platform that allows anyone to assemble statistical/numerical methods and data on the server (an Elastic-R virtual machine instance on the cloud) and to visually create and publish, in the form of URLs, interactive user interfaces and dashboards exposing those methods and data. Elastic-R also provides tools that allow anyone to expose those methods (implemented by R functions for example) as SOAP Web services that can be used as computational services on the cloud for data analysis pipelines or as nodes for workflow workbenches.

Recording capabilities: Because the scientific computing environments accessible through Elastic-R are running on virtual machines and because the working directories are hosted by virtual disks, a snapshot of the full updated computational environment can be produced at any point in time. That snapshot can be archived or made available to anyone using the Elastic-R workbenches: an author can share his environment with the reviewers of the journal to which he submitted his paper, a teacher can make the statistical learning environment needed for his course available to his students, a researcher in laboratory A can make his simulation environment accessible to his collaborators in laboratory B, etc.

This chapter is organized as follows: The first section describes the building blocks of the Elastic-R platform and the ecosystem it creates for the interoperability, sharing and reuse of the computing artifacts. The second section describes how the usage scenarios of Elastic-R can be integrated with those of an IaaS. The third section details the major e-Science use cases Elastic-R deals with. The fourth section shows how it can be used as a highly productive cloud applications platform.

2. An Open Platform for Scientific Computing, the Building Blocks.

R is a language and environment for statistical computing and graphics and it became the *lingua franca* of data analysis [5][7]. R has a very powerful graphics

system as well as cross-platform capabilities for packaging any computational code. Hundreds of available R packages, exponentially growing in number, implement the most up-to-date computational methods and reflect the state-of-the-art of research in various fields. R packages have become a reproducible research enabler because they enable functions and algorithms to be reused and shared. There is no obstacle to a large-scale deployment of R on public clouds and Grids since it is licensed under the GNU GPL [11]. However, R is not multithreaded and does not operate as a server. As a language, it implements the powerful S4 class system but as a library, R has only a low-level non-object oriented Application Programming Interface (API). Graphical User Interfaces (GUIs) development for R remains non standardized. R's potential as a computational back end engine for applications and service-oriented architectures has yet to be fully exploited. While its user base is growing at a high rate, this growth rate would be significantly higher with a user-friendly and rich workbench. Elastic-R brings to the R ecosystem all those missing features which may enable it to be applied in many more situations, in various different ways. Its ambition though goes far beyond the provision of new tools and frameworks. By extending R's logic of openness and extensibility, Elastic-R builds an environment where all the artifacts and resources of computing become "pluggable" and not only the computational component (the R package).

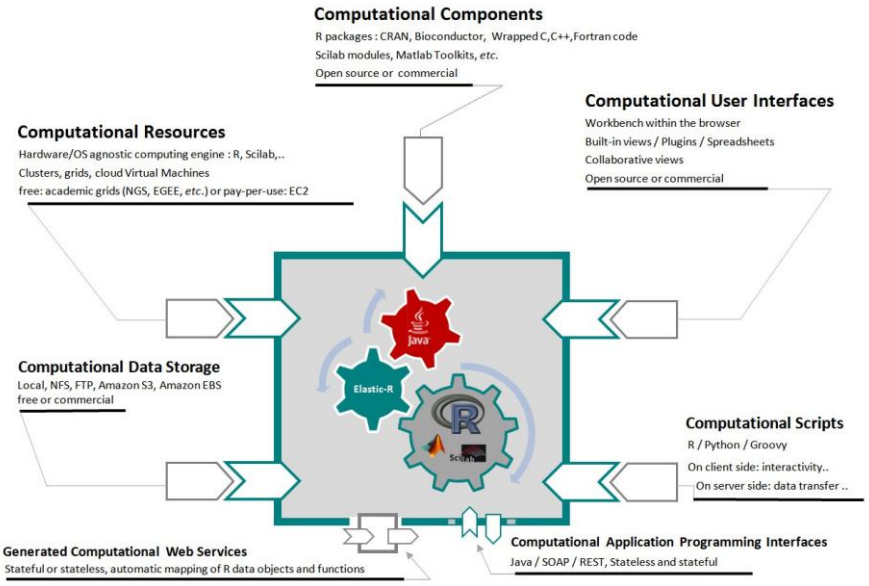

Figure 1. The open computational platform's ecosystem.

Figure 1 shows the key features of Elastic-R. The Java or Ajax Elastic-R workbench allows the scientist, the statistician, the financial analyst, etc. to easily

assemble (plug together) synergetic capabilities, described in the following subsections:

2.1 The processing capability

By providing a simple URL and credentials, the scientist connects his workbench to an R-based remote computational engine and gain access to the computational resource whether it is a node of Grid, a virtual machine of a cloud, a cluster or his own laptop. The engine is agnostic to its hosting operating system and hardware. IaaS requires such a mechanism for computing to become "like electricity". EC2 [22] allows the user to choose the capacity of the virtual machine (number of virtual cores, memory size, etc.) he would like to launch. The Elastic-R workbench exposes that choice to the scientist in its simplified EC2 Console. The state of an Elastic-R engine persists until the computational resource is released (the virtual machine is shut down, the interactive Grid job is killed, the process on the physical server is killed, etc.). The scientist can disconnect form the engine and reconnect again from anywhere: he retrieves his session with all variables, functions, graphics, spreadsheets, etc.

2.2 The mathematical and numerical capability

By gaining access to an R session and by importing into his workspace the R packages related to his problem domain, the scientist gathers the functions and mathematical models needed to process data and transform it into knowledge and insight. The R package can be a wrapper of any mathematical library written in C, C++, FORTRAN, etc. R can be considered as a universal framework for computational code and computational toolkits. From within his R session, the scientist can also call Scilab [6], Sage [13], Root [14], etc. and increase the mathematical capability of his environment. An architecture for server-side extensions allows anyone to build java bridges that couple the Elastic-R engine with any software. Such bridges are available for Matlab [16] and OpenOffice [15].

2.3 The orchestration capability

The S language implemented by R is one of the most powerful languages ever created for "programming with data" [12]. Besides R, the user can orchestrate tasks and control data flow using python and groovy. Interpreters for theses scripting languages are embedded both within the Elastic-R engine (on server side) and within the workbench (client side). The full capabilities of the platform are exposed via SOAP and RESTful front-ends (Computational Application Programming Interfaces in figure 1) and the Elastic-R engine can be piloted programmatically from Java, Perl, C#, C++, etc. A tool is provided to enable the scientist to generate and deploy SOAP Web services exposing a selection of his R

functions (Figure 1: generated computational web services). They can be used as nodes within workflow workbenches. The nodes are dynamically connected to the scientist's R session and the processing of data is done on the cloud if the Elastic-R engine exposing the Web service is on the cloud.

2.4 The interaction capability

The console views within the Java and Ajax workbenches allow the full control of an R session as well as the use of python, groovy and Linux shells. Besides the consoles, both workbenches have several dockable built-in views including remote directory browsers for the viewing, download and upload of files from and to the remote engine's working directory, syntax-highlighting-enabled code editors, help browsers, viewers for various file formats (PDF, SVG, HTML, etc.), interactive server-side graphic devices with built-in resizing, zooming, scrolling, coordinate tracking and annotation capabilities, data inspectors, linked plots, spreadsheets fully integrated with R functions and data.
The workbench's architecture for plugins lets anyone create his own views and dashboards to make workbenches more productive or to expose statistical and numerical models through simple graphical user interfaces. All the views of the workbenches are collaborative: when more than one user is connected to the same Elastic-R engine, the actions of one collaborator are broadcasted to all the others. An example of the Elastic-R Java workbench is shown in Figure 2.

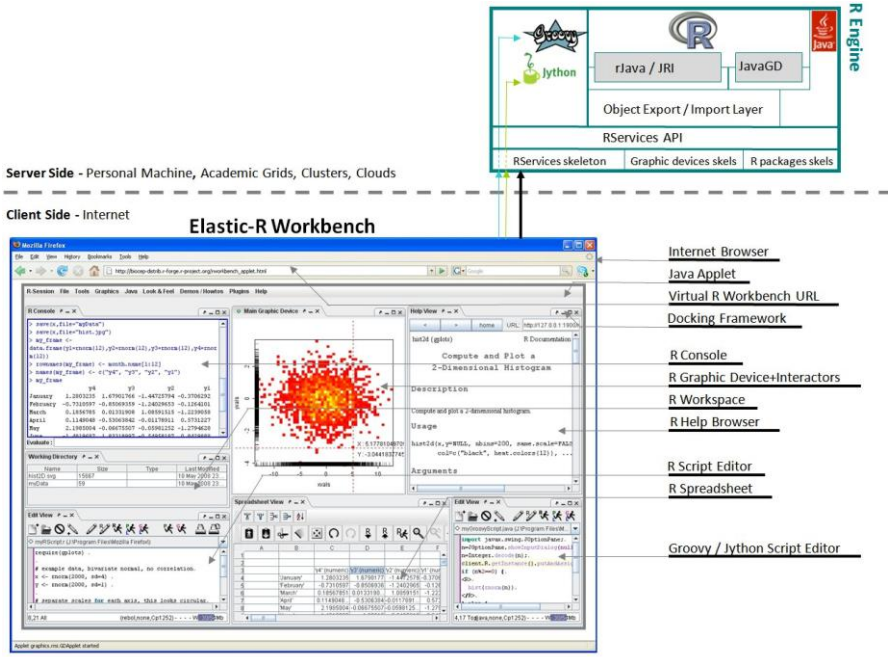

Figure 2. Elastic-R Java workbench.

2.5 The persistence capability

The Elastic-R computational engine's working directory can be on a local or a network file system and its content can be easily synchronized with an FTP server or with Amazon S3 [21]. When the scientist uses the simplified EC2 console to start a Elastic-R-enabled Amazon Machine Image (AMI), an Amazon Elastic Block Store (EBS) is automatically attached to the running AMI. That EBS becomes the working directory of all the Elastic-R engines hosted by the AMI and all the files generated by the scientist including the workspaces serialization, the spreadsheets content, the generated web services, etc. are kept when the AMI is shutdown. The EBS is also a place where the R packages installed by the scientist are stored. Those packages are made available to the Elastic-R engines when a new AMI is started. A snapshot of the EBS can be created by the scientist who can decide to share that snapshot with other EC2 users.

3. Elastic-R and Infrastructure-as-a-service

Elastic-R can be used on any type of infrastructure. However, the platform takes its full dimension only when it is used within an IaaS-style-cloud whether it is Amazon EC2 or a private cloud based on Eucalyptus [17], OpenNebula [18] or Nimbus [19].

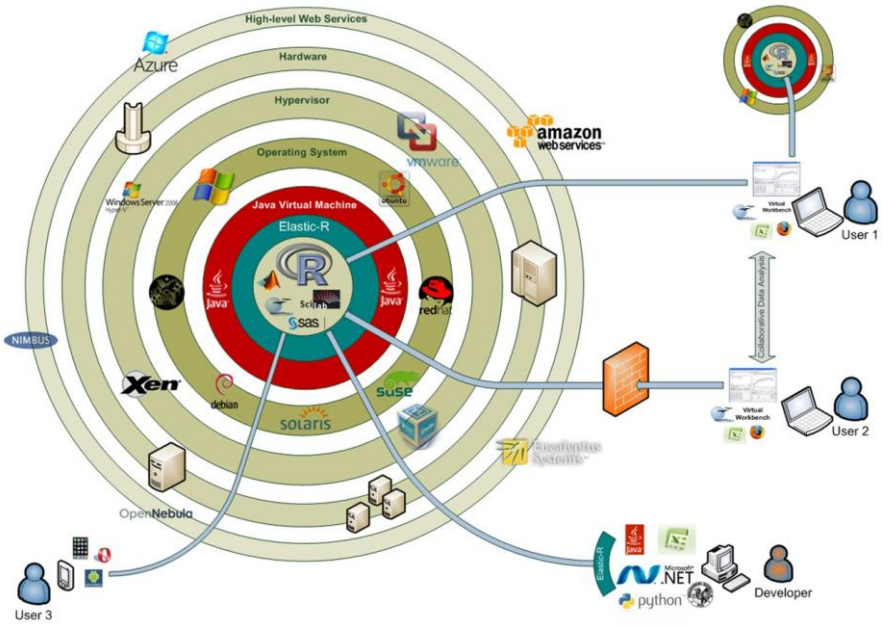

Figure 3. Elastic-R in the IaaS environment.

Figure3 shows the role of Elastic-R within an Infrastructure as service environment (the concentric circles). Elastic-R wraps R (in the very centre) with Java Object-oriented layers that can be accessed remotely from anywhere. The R engine created (inside the Java virtual machine circle) is agnostic to the operating system and to the hardware. In this case, it runs within a virtual machine that can be based on any OS. The virtual machine uses the resources of the hardware via the Hypervisor and its management by the end user (start-up, shutdown, etc.) is done using the outer layer (IaaS API). "User1" and "User2" can connect to the R engine and use it collaboratively from their Java workbenches, their Ajax workbenches or from their Excel spreadsheets. The Developer can use the R engine (one or many, on one or many virtual machines) by calling its remote API (Java-RMI, SOAP, REST) from his ASP.NET web application, his Java desktop application, his Perl scripts, his Excel Add-in, etc. "User 3" from his portable device (iPhone, Android-based phone, etc.) can use the Ajax workbench to access the same R engine as "User 1" and "User 2" and show them his data, spreadsheets, slides, etc. interactively: the Ajax workbench gives the same capabilities to "User 1", "User 2" and "User 3". They all can issue commands to R, install and use new packages, generate and interact with graphics, upload and process files, download results, etc.

3.1 The Building Blocks of a Traceable and Reproducible Computational Research Platform

Elastic-R on an IaaS-style cloud provides 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. Elastic-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 Elastic-R Java and Ajax workbenches make it possible to all scientists to work with these snapshots (AMIs + EBSs) and produce them easily. By Providing the Elastic-R-enabled 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.

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

Besides being free and mostly open source and therefore accessible to students and educators, Elastic-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 Elastic-R, it becomes possible for educators to hide the complexity of R, Scilab, Matlab, etc. with User Interfaces such as the Elastic-R plugins and 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.

Educators can adapt the Elastic-R virtual machines images to the specific needs of their courses and tutorials. For example, after choosing the most appropriate image, they can add to it the missing R packages, the required data files, install the missing tools, etc. The new image can then be provided to students on USB keys or made accessible on an IaaS-style cloud. In the first case, 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 Elastic-R workbench and to connect to a computational engine on the virtual machine. In the second case, they need only a browser. Once again, a virtual machine prepared by one educator can be shared, reused and improved by other educators.

The virtual machine is fully self-contained: the code needed to run the workbench or the plug-ins prepared by the educator can be delivered by the virtual appliance itself thanks to the Elastic-R code server that runs at startup. The interaction between the student and the SCE as well as the artifacts produced are saved within the Elastic-R-enabled-virtual machine. The educator can retrieve the USB keys used by the students (or connect to the virtual machine instance on the IaaS-style cloud) 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 workbench open also new perspectives in distributed learning. The educator can connect anytime to the SCEs of students at any location. He can see and update their environments and guide them remotely. Collaborative problem solving becomes also possible and can be used as a support for learning.

4. Elastic-R, an e-Science Enabler

Elastic-R is an e-Science platform that deals with some of the most timely use cases related to the use of Information and Communications Technologies (ICT) in research and education:

4.1 Lowering the Barriers for Accessing on-Demand Computing Infrastructures. Local/Remote Transparency

The same application, the Elastic-R 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. Switching from one resource to another (for example from one virtual machine instance on Amazon Elastic Compute Cloud to another or from an interactive Grid job on the European Grid EGI to an interactive job on an intranet cluster) becomes as simple as replacing one URL with another.

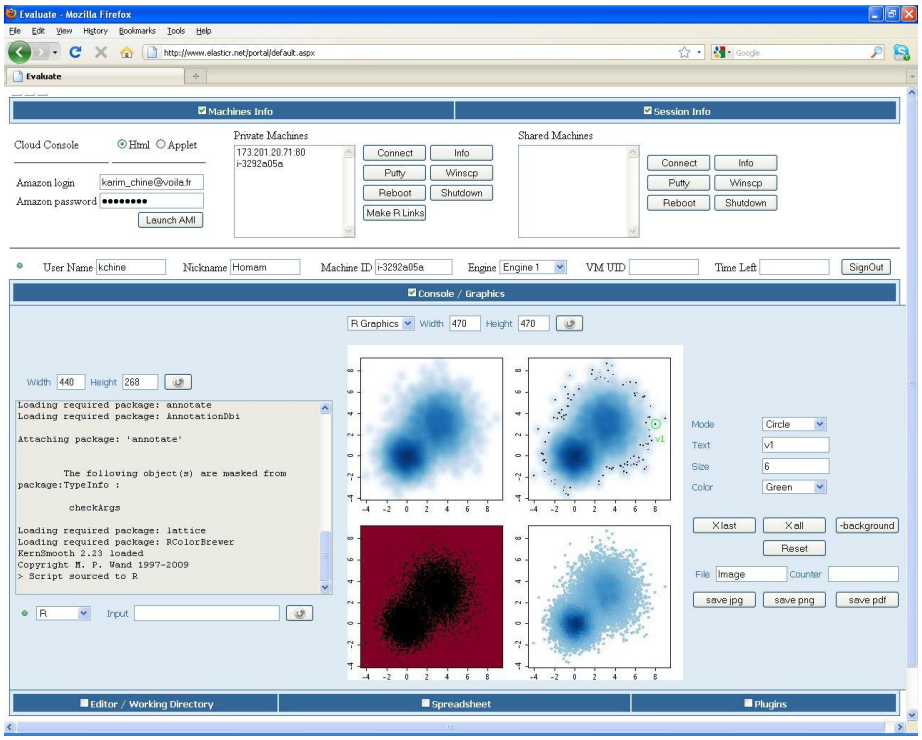

Figure 4. Elastic-R Ajax workbench.

4.2 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 Elastic-R enables 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. This is the case when an Elastic-R EC2 AMI is used to process data that is already on Amazon's Elastic Cloud. The

user can connect his virtual workbench (or his scripts using the Elastic-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.

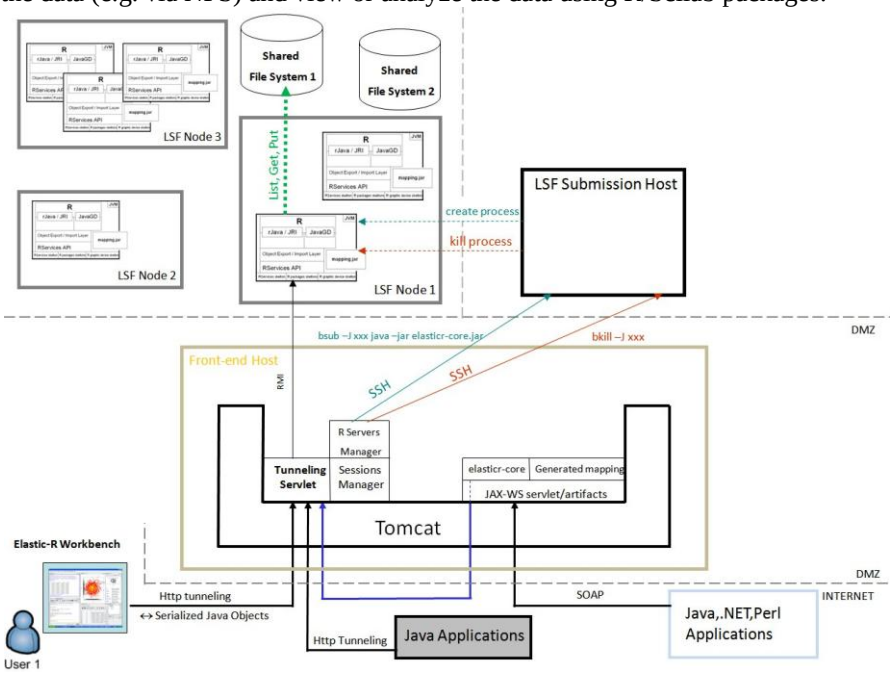

Figure 5. Elastic-R on an LSF cluster.

4.3 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. 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. Linked plots views based on a refactored iplots package [8] enable collaborative highlighting and color brushing on a variety of high interaction graphics.

4.4 Science Gateways Made Easy

Web-based interfaces and portals allowing scientists to use federated distributed computing infrastructures 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. Elastic-R proposes a different paradigm for the creation and

distribution of such front-ends to HPC/cloud environments with plugins and server-side spreadsheets (see 5.1 and 5.2)

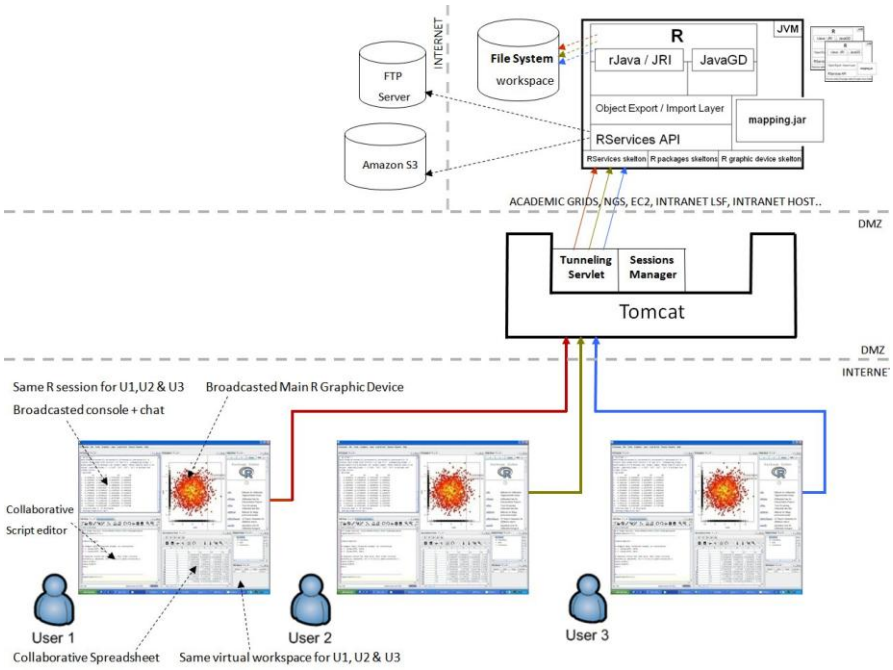

Figure 6. Collaborative session with the Elastic-R workbench.

4.5 Bridging the Gap between Existing Scientific Computing Environments 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, OpenOffice Calc or Excel to access and use the Grid/cloud-enabled engine. For example, an Excel add-in enables scientists to use the full capabilities of the Elastic-R platform and reproduce the features of the Elastic-R spreadsheets from within Excel. The bi-directional mirroring of server-side spreadsheets' models into Excel cell ranges is also available. This allows 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.

4.6 Bridging the Gap between Mainstream Scientific Computing Environments

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 can be integrated (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 SOAP API can be called from any environment. It enables SciPy users for example to work with Elastic-R engines on the cloud and to call R and Scilab functions.

4.7 Bridging the Gap between Mainstream Scientific Computing Environments and Workflow Workbenches

Elastic-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 [24], Taverna [25] or Pipeline Pilot [26]. 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.

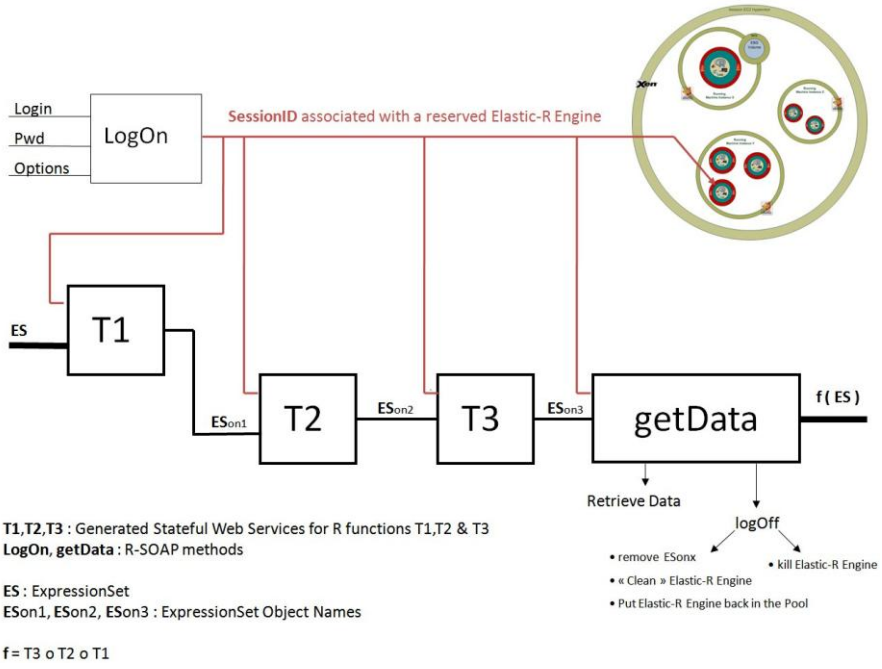

Figure 7. Workflows with generated Stateful SOAP Web services.

4.8 A Universal Computing Toolkit for Scientific Applications

Elastic-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.)

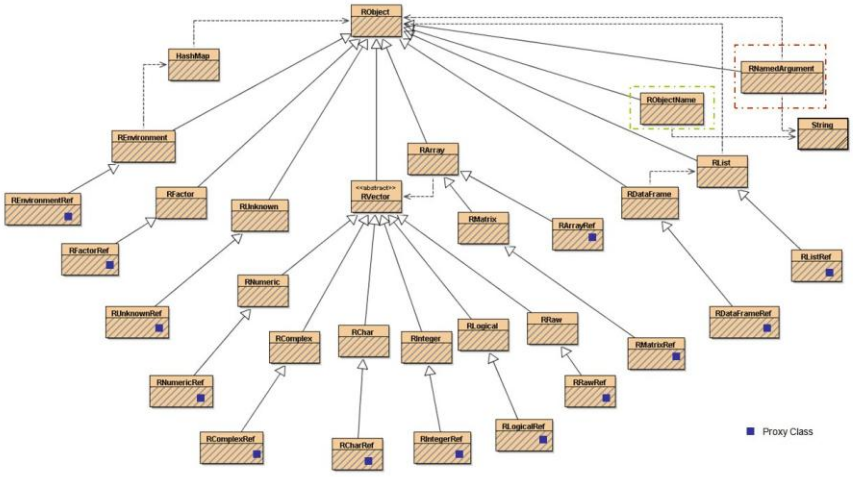

Figure 8. Java classes diagram: mapping of standard R objects

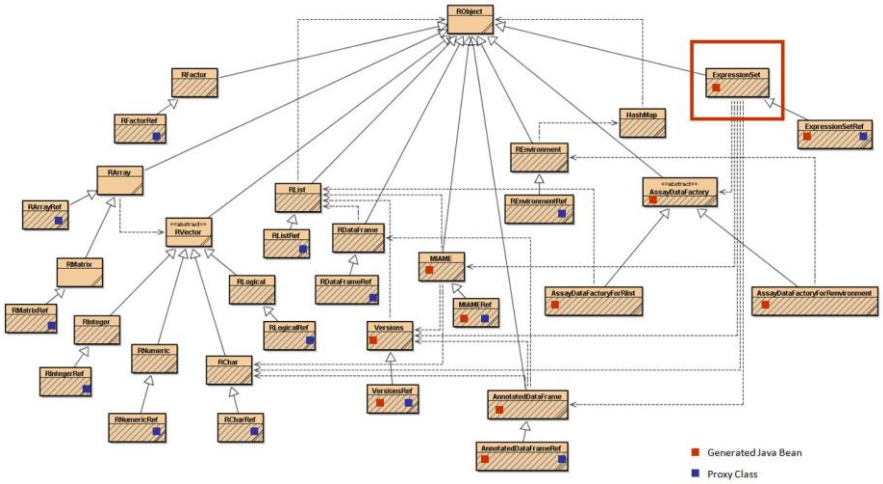

Figure 9. Java classes diagram: generated mapping for ExpressionSet (S4 class)

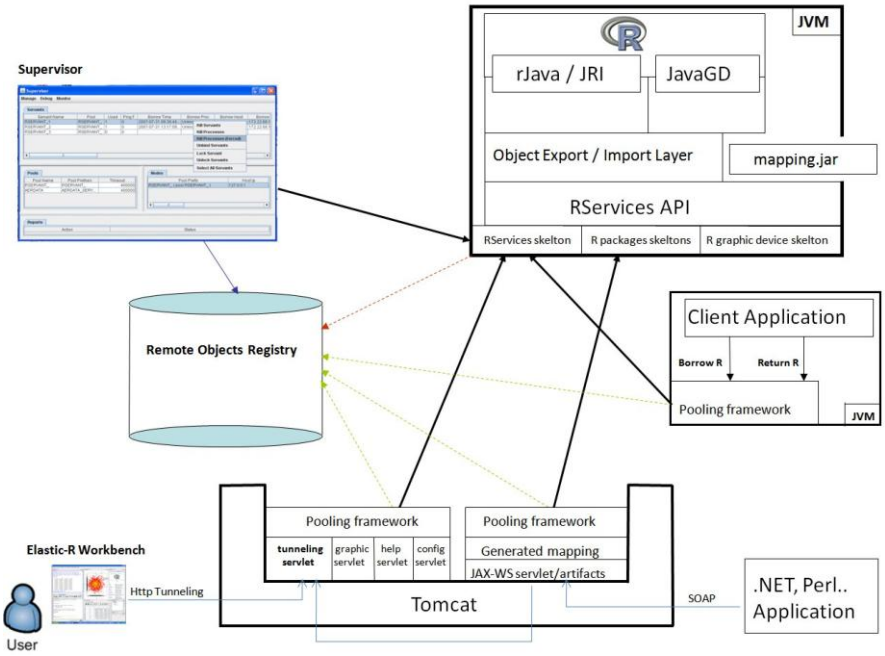

Figure 10. Elastic-R engine - usage scenarios and architecture.

4.9 Scalability for Computational Back-Ends

Elastic-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.

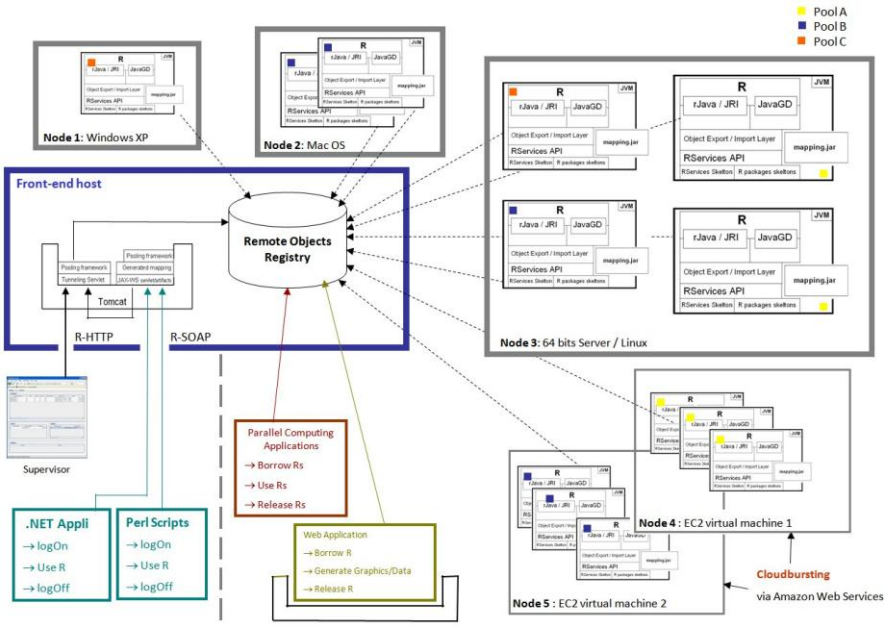

Figure 11. Elastic-R engines pools, usage scenarios and architecture.

4.10 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. Elastic-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.

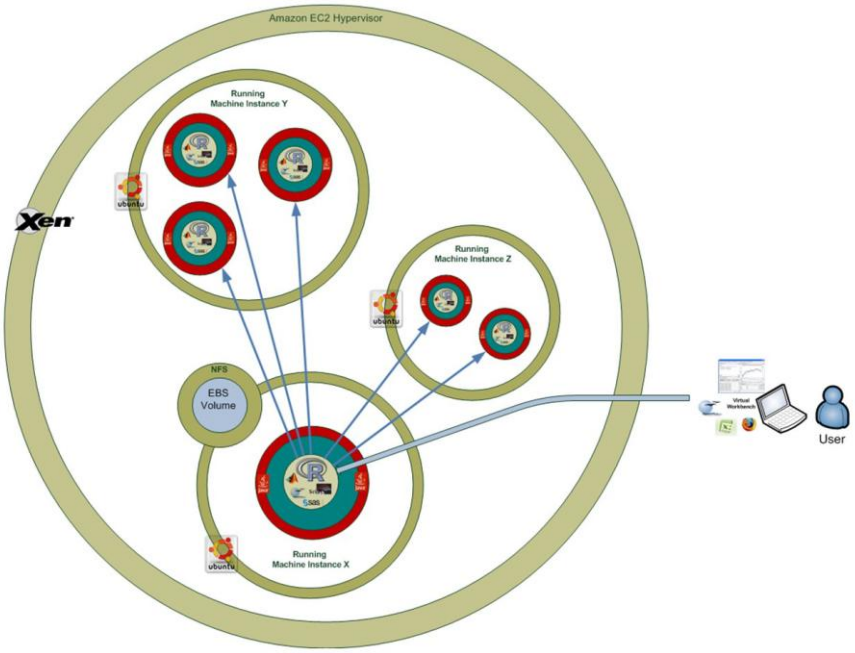

Figure 12. Parallel computing with Elastic-R on Amazon Elastic Compute Cloud.

5. Elastic-R, an application platform for the Cloud

Elastic-R is extensible with java components both on client-side (the plugins) and on server-side (the extensions). With those components, anyone can create and deploy his application on the cloud without having any specific knowledge about the infrastructure.

5.1 The Elastic-R Plug-ins

The Elastic-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 Elastic-R 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 (Elastic-R's Java Web Start trigger + computational engine's URL parameter + plugin'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.

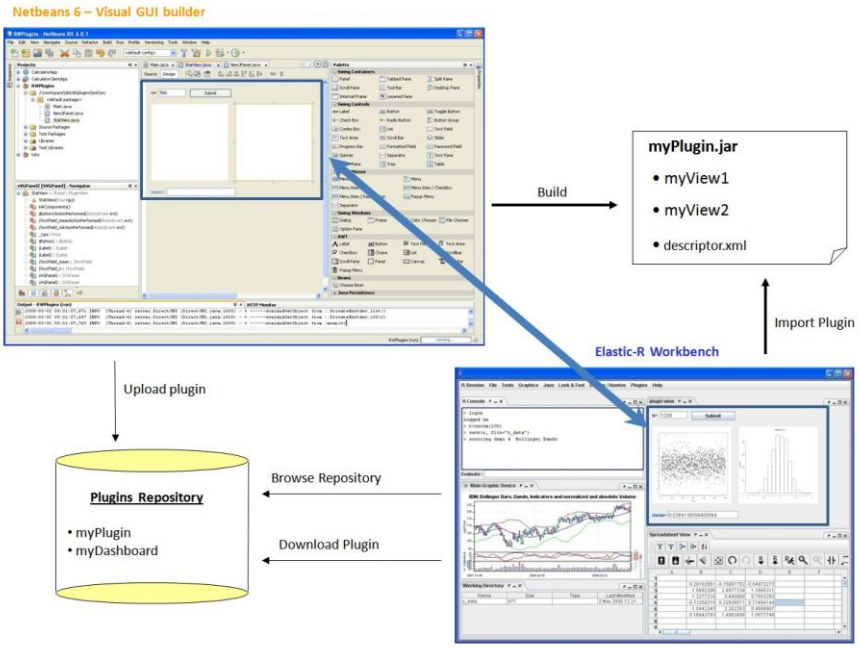

Figure 13. Elastic-R plugins' visual creation and publishing.

5.2 The Elastic-R Spreadsheets

The Elastic-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 the spreadsheet entirely. 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 or cloud-based analytical views and dashboards and lowers the barriers for creating science gateways and distributing them.

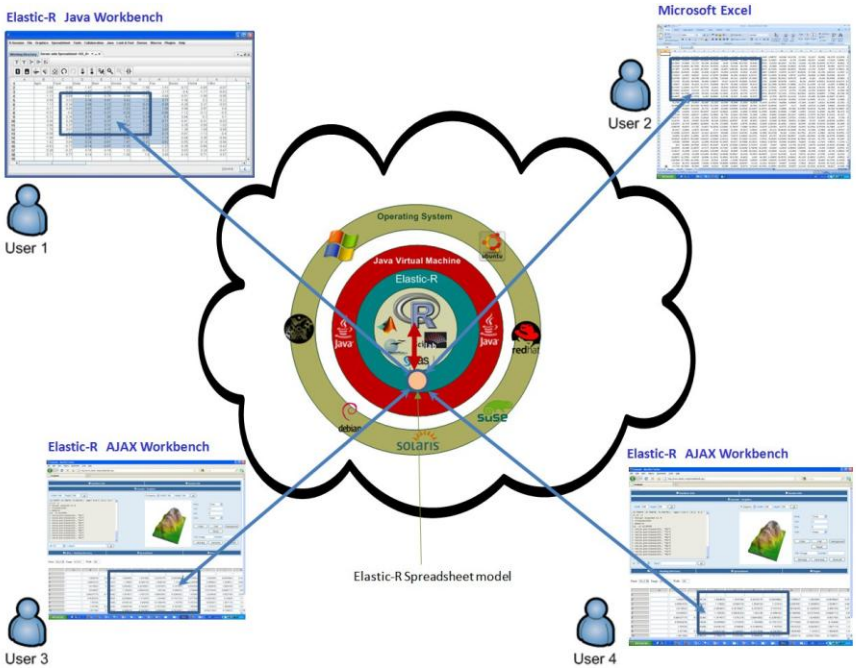

Figure 14. Elastic-R spreadsheets

5.3 The Elastic-R extensions

Elastic-R extensions are Java components that can be uploaded anytime to an Elastic-R engine's extensions folder. They are dynamically loaded by the engine's Java Virtual Machine and the code they expose can be called from the client. Extensions allow anyone to build java bridges that couple the Elastic-R engine with any software such as Matlab [16] and OpenOffice [15].

6. Conclusions and Future Directions

This article described the Elastic-R as a new environment that has the potential to democratize the cloud and to push forward the reproducibility of computational research. Its current availability and easy access on Amazon Elastic Compute Cloud maximizes 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 [10] for on demand computing is being

built using the different frameworks provided by Elastic-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, VMware, EC2, R and Elastic-R prove that the target of a universal computational environment for science and for everyone is definitely within reach.

References

1. http://en.wikipedia.org/wiki/Cyberinfrastructure.
2. I. Foster. What is the Grid? A Three Point Checklist. Grid Today, 1(6):22-25, 2002.
3. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the Grid. Grid Computing: Making the Global Infrastructure a Reality, 2003.
4. Shantenu Jha, Andre Merzky, Geoffrey Fox, "Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes"
5. 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.
6. http://www.scilab.org
7. http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html
8. Theus, M. and Urbanek, S. (2008) Interactive Graphics for Data Analysis: Principles and Examples, CRC Press, ISBN 978-1-5848-8594-8
9. www.elasticr.net/platform
10. www.elasticr.net/portal
11. Programming with Data: A Guide to the S Language, John M. Chambers, Springer-Verlag, New York, 1998.
12. http://www.sagemath.org/
13. http://root.cern.ch/
14. http://www.openoffice.org/
15. http://www.mathworks.co.uk/
16. http://open.eucalyptus.com/
17. http://www.opennebula.org/
18. http://www.nimbusproject.org/
19. Science Clouds: Early Experiences in Cloud Computing for Scientific Applications, Keahey K., T. Freeman. Cloud Computing and Its Applications 2008 (CCA-08), Chicago, IL. October 2008
20. Interview with Kate Keahey, "An interview with Kate Keahey of the Nimbus project, a cloud computing infrastructure" [Online]. Available: http://www.nsf.gov/news/news_videos.jsp?cntn_id=114788&media_id=65105
21. Amazon, Inc., "Amazon Simple Storage Service." [Online]. Available: aws.amazon.com/s3
22. Amazon, Inc., "Amazon Elastic Compute Cloud." [Online]. Available: aws.amazon.com/ec2
23. NetSolve to GridSolve: The Evolution of a Network Enabled Solver, Asim YarKhan, Jack Dongarra, and Keith Seymour, IFIP WoCo9 conference "Grid-Based Problem Solving Environments: Implications for Development and Deployment of Numerical Software", Prescott, AZ, July 17-21, 2006.
24. http://www.knime.org/
25. http://www.taverna.org.uk/
26. http://accelrys.com/products/scitegic/

Index terms (alphabetically):

Ajax
Android
EC2
Grid
IaaS
iPhone
Java
Open Science
Parallel Computing
Portal
R
Reproducible research
Scilab
SOAP
S
S3
Scientific Computing Environment
Statistical Computing
Virtualization
VM ware
Workbench
Workflow