RBioCloud: A Light-weight Framework for Bioconductor and R-based Jobs on the Cloud

Ishan Patel
IBM Canada Ltd., Bedford, Canada
Email: ishanp@ca.ibm.com

Blesson Varghese1 and Adam Barker
School of Computer Science, University of St Andrews, UK
Email: {varghese, adam.barker}@st-andrews.ac.uk

Abstract—Large-scale ad hoc analytics of genomic data is popular using the R-programming language supported by 671 software packages provided by Bioconductor. More recently, analytical jobs are benefiting from on-demand computing and storage, their scalability and their low maintenance cost, all of which are offered by the cloud. While Biologists and Bioinformaticists can take an analytical job and execute it on their personal workstations, it remains challenging to seamlessly execute the job on the cloud infrastructure without extensive knowledge of the cloud dashboard. How analytical jobs can not only with minimum effort be executed on the cloud, but also how both the resources and data required by the job can be managed is explored in this paper. An open-source light-weight framework for executing R-scripts using Bioconductor packages, referred to as ‘RBioCloud’, is designed and developed. RBioCloud offers a set of simple command-line tools for managing the cloud resources, the data and the execution of the job. Two biological test cases validate the feasibility of RBioCloud. The framework is publicly available from http://www.rbiocloud.com.

Keywords—Cloud computing, R programming, Bioconductor, Amazon Web Services, Data analytics

I. INTRODUCTION

Ad-hoc analytics of genomic data is popular in domains such as computational biology and bioinformatics. Typically, an analytical job comprises software scripts written by biologists or bioinformaticists in high-level programming languages, such as R [1], along with large amounts of data that needs to be processed. R-based analytics in computational biology or bioinformatics is gaining popularity and is supported through 671 software packages provided by Bioconductor [2].

Analytical jobs which may require a few hours or perhaps even a few days may ingest large amounts of data and subsequently also produce data in large volumes. The cloud has become an appealing platform for analytics since it offers on-demand computing and storage resources, along with scalability and low maintenance costs [3]. This has led to a variety of research for supporting computational biology and bioinformatics related jobs on the cloud (for example, genome sequencing [5] and biomedical computing [6]).

Software projects such as elasticR [7] and AzureBlast [8] support applications on the cloud, all of which require the user to have extensive knowledge of the cloud dashboard to be able to port an existing analytical workload onto the cloud. The options provided in projects for a fully configurable cloud cluster can fit well with the skill set of a cloud developer, thereby narrowing their wide usage. Moreover, adapting projects to execute workloads developed using the R programming language is cumbersome, specific adaptations being required in many cases. A similar challenge exists for executing the increasing number of analytical workloads that are developed using the R with Bioconductor packages [9].

The current Bioconductor based solution [10] is based on manually configuring the cloud dashboard for every job that needs to be executed. Software developed to support R and Bioconductor, for example, Myrna [11] and Contrail1 are restricted to specific applications in computational biology and bioinformatics. These challenges can be overcome by the development of a generic framework that can support R-based jobs supported by Bioconductor packages, and their execution and management on the cloud.

The research reported in this paper aims to address the above challenges. A light-weight framework, ‘RBioCloud’, for supporting R-based analytical applications which use Bioconductor software packages and need to be executed on the cloud is presented. Domain scientists have to often spend a lot of time dealing with the complex details of configuring the cloud. Using RBioCloud, an analytical job can be executed on the cloud with minimal effort using a set of five commands from a personal workstation. The need for any extensive knowledge of the cloud dashboard is minimised.

The remainder of this paper is organised as follows. Section II considers the design of the RBioCloud framework. Section III describes the command line tools offered by RBioCloud for managing and executing an analytical job. Section IV presents two test cases to validate the feasibility of RBioCloud. Section V concludes this paper by considering future work.

II. FRAMEWORK DESIGN

Figure 1a, shows the design of the RBioCloud framework which is located on a host site for accessing and managing cloud resources. The host site represents the workstation of a computational biologist or a bioinformaticist who makes use of the cloud infrastructure to execute a job. The Amazon cloud infrastructure is employed in this research. RBioCloud is designed so that the job can be executed from the host site using the following five step sequence (refer Figure 1b):

- Step 1: Gather resources - initialise cloud compute and storage resources from the host.
- **Step 2**: Submit job - send the analytical job from the host onto cloud resources.
- **Step 3**: Execute job - execute the scripts within the job on the resources.
- **Step 4**: Retrieve results - get results generated on the cloud resources onto host.
- **Step 5**: Terminate resources - release all resources which were initialised on the cloud.

RBioCloud is developed using the Python programming language and is supported by a number of interfaces. The compute and storage resources are provided by the Amazon Web Services (AWS)\(^2\). All resources are available on-demand and are paid for on the basis of their usage. The computational resources are offered through Elastic Compute Cloud (EC2)\(^3\) and are available as instances. The storage resources are referred to as the Elastic Block Storage (EBS)\(^4\) provide persistent data storage. Two Python interfaces, namely BOTO\(^5\) provides the interface to access the resources provided by AWS and Fabric\(^6\) facilitates remote administration of the cloud resources.

Amazon instances are initialized using Amazon Machine Images (AMI)\(^7\). The RBioCloud framework is built on the Bioconductor Cloud AMI \([10]\) and supports the R programming language along with Bioconductor packages. The cloud is attractive for large analytical jobs as parallel computations incorporated within jobs can be exploited on the cloud. The Simple Network Of Workstations (SNOW)\(^8\) interface is employed for parallel execution of jobs on the cloud.

### III. Tools

The five command line tools offered by RBioCloud to support gathering of cloud resources, to submit and execute a job, retrieve results from the cloud and terminate resources are presented in this section.

<table>
<thead>
<tr>
<th>Sequence of Activities</th>
<th>Step 1: Gather Resource</th>
<th>Step 2: Submit Job</th>
<th>Step 3: Execute Job</th>
<th>Step 4: Retrieve Results</th>
<th>Step 5: Terminate Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST SITE</td>
<td>RBC_GatherResource</td>
<td>RBC_SubmitJob</td>
<td>RBC_ExecuteJob</td>
<td>RBC_GetResults</td>
<td>RBC_TerminateResource</td>
</tr>
<tr>
<td>AMAZON CLOUD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RBC_GatherResource** provisions configuring an instance or multiple instances and a cluster on the cloud. The syntax of the command is:

```
```

A job comprises the script that needs to be executed and the data required by the script both of which are submitted to the cloud using the `rsync` protocol. One advantage of using rsync is that subsequent data transfers are quickly synchronised between the host and the cloud. The submission of a job is facilitated using **RBC_SubmitJob** and the syntax is:

```
RBC_SubmitJob [-h] [-v] [-rname RESOURCE_NAME [-toallnodes | -tomaster]] [-jobdir JOB_DIRECTORY] [-data]
```

**RBC_ExecuteJob** executes a job on the cloud resource and the syntax of the command is:

```
```

**RBC_GetResults** retrieves results from the cloud resource onto the host and the syntax is:

```
RBC_GetResults [-h] [-v] [-rname RESOURCE_NAME [-frommaster | -fromall]] [-jobdir JOB_DIRECTORY] [-runname RUN_NAME]
```

After the completion of a job, the resources on the cloud need to be released using **RBC_TerminateResource**. The syntax is:

```
RBC_TerminateResource [-h] [-v] [-rname RESOURCE_NAME] [-deletevol]
```

### IV. Feasibility Study

Popular biological jobs include searching, analysing and normalising data \([12]\). Two test cases that represent such biological jobs are selected to demonstrate the feasibility of RBioCloud. Firstly, genome searching, and secondly, normalisation of microRNA (miRNA) microarray data are presented.
A. Test case 1: Genome searching on an Instance

The first test case is based on the BSgenome software package [13], and the script executed is GenomeSearching.R which performs efficient genome searching with Biostrings and BSGenome data packages. The R script loads BSgenome.Celegans.UCSC.ce2, which is the ce2 genome for chromosome I of Caenorhabditis elegans [14]. The script finds an arbitrary nucleotide pattern in a chromosome and in an entire genome. For executing the script using RBioCloud, the job is organised into one directory, for example BSGenome, which contains the GenomeSearching.R script and all associated data. BSGenome also needs to contain two additional directories Results and RunResults (a similar directory structure needs to be followed for executing any job using RBioCloud). All the results that need to be generated by the script need to be directed to Results. RunResults is not submitted onto the cloud but remains on the host site to retrieve and store results of each individual run. The following sequence of five commands will execute GenomeSearching.R on the cloud and fetch the results onto the host site:

1 > RBC_GatherResource -rname 'BSgenome_instance' -rsize 1 -desc 'For_Genome_Searching'
2 > RBC_SubmitJob -rname 'BSgenome_instance'
3 > RBC_ExecuteJob -rname 'BSgenome_instance' -rscript 'GenomeSearching.R' -runname 'Run1_on_BSgenome_instance'
4 > RBC_GetResults -rname 'BSgenome_instance' -runname 'Run1_on_BSgenome_instance'
5 > RBC_TerminateResource -rname 'BSgenome_instance' -deletevol

When the first command of the sequence is executed one EC2 instance is initialised using the Bioconductor AMI, and tagged as BSgenome_instance. If optional arguments such as type of instance and EBS volume are not provided then the default values which are defined in the RBioCloud configuration file are chosen; the default values can be edited. The BSgenome folder is synchronised with BSgenome_instance when the second command is executed; BSGenome is the current working directory from which the RBC_SubmitJob is executed. The script, GenomeSearching.R from BSGenome directory is executed on BSgenome_instance with a run name, Run1_on_BSgenome_instance, when the third command is executed. The results from Run1_on_BSgenome_instance are retrieved on to the host Results directory when the fourth command is executed. The Amazon resource BSgenome_instance is terminated using the fifth command. The multiple execution of the RBC_GatherResource command facilitates the creation of multiple instances, and multiple instances cannot have the same name.

The job is to find nucleotide patterns in an entire genome and produce results. The input is a dictionary, containing 50 patterns, each of which is a short nucleotide sequence of 15 to 25 bases. The forward and reverse strands of seven Caenorhabditis elegans chromosomes named as chrI, chrII, chrIII, chrIV, chrV, chrX, chrM are the target. Results generated by the script need to be directed to Results (any job using RBioCloud). All the results that need to be generated by the script need to be directed to Results. RunResults is not submitted onto the cloud but remains on the host site to retrieve and store results of each individual run.

B. Test case 2: Normalisation of microRNA (miRNA) microarray data on a Cluster

The second workflow is based on the LVSmiRNA software package [15]. The script executed is LVSmiRNA.R which normalises microRNA (miRNA) microarray data. The Least-Variant Set (LVS) normalisation method [16] is employed in the package and the input is the miRNA expression data provided as Comparison_Array.txt. The script then identifies a subset of miRNAs with the smallest array-to-array variation, using the estVC function. The first result obtained from the script is an RA-plot, which is a scatter plot (refer Figure 3a) with logarithmic scales showing the array effect versus standard deviation. The second result obtained from the script is a box plot (refer Figure 3b) of data after normalisation.

The estVC function can benefit from using parallel computation for achieving higher speed up over sequential computation, and can take a cluster object as an argument. Here Amazon clusters can come to play, and will need to be manually configured using the Amazon dashboard as shown in [10]. Employing RBioCloud will be easier as the user can configure this as a single parameter in the RBC_GatherResource command.

To execute the LVSmiRNA.R script on an Amazon cluster, the script and the input data needs to be provided in a directory,
for example LVSmiRNA, and the directory also needs to contain two additional sub-directories Results and RunResults. The two graphs generated by the script needs to be directed to Results, RunResults is not submitted onto the cloud but remains on the host site to store results of every individual run. The following sequence of five commands will execute LVSmiRNA.R on a cloud cluster and fetch the results onto the host site:

1. `RBC_GatherResource -rname 'LVSmiRNA_cluster' -rsize 8 -desc 'For_LVS_miRNA`
2. `RBC_SubmitJob -rname 'LVSmiRNA_cluster'`
3. `RBC_ExecuteJob -rname 'LVSmiRNA_cluster' -rscript 'LVSmiRNA.R' -runname 'Run2_on_LVSmiRNA_cluster'`
4. `RBC_GetResults -rname 'LVSmiRNA_cluster' -runname 'Run2_on_LVSmiRNA_cluster' -frommaster`
5. `RBC_TerminateResource -rname 'LVSmiRNA_cluster' -deletevol`

A cluster with eight EC2 instances is initialised using the Bioconductor AMI, and tagged as LVSmiRNA_cluster when the first command is executed. Should the optional arguments such as type of instance and EBS volume be not provided then the default values which are defined in a configuration file are chosen. The LVSmiRNA folder is synchronised on LVSmiRNA_cluster when the second command is executed; LVSmiRNA is the current working directory. The script, LVSmiRNA.R from LVSmiRNA is executed on LVSmiRNA_cluster with a run name, Run2_on_LVSmiRNA_cluster when the third command is executed. The resultant graphs from Run2_on_LVSmiRNA_cluster run are retrieved on to the host Results directory when the fourth command is executed. The Amazon resource LVSmiRNA_cluster is terminated using the fifth command.

V. CONCLUSIONS

Gathering and managing vast cloud resources in the computational biology or bioinformatics setting for executing an analytical job can be cumbersome. This is not because cloud resources aren’t readily accessible, but the pipeline for executing an analytical job on the cloud requires extensive knowledge of the cloud. While high-performance computer architects may be able to design and deploy such workflows for production based applications it may not be easily possible for biologists with limited high-performance computing skills to perform ad hoc analytics. To allow analytical jobs to fully benefit from the cloud there needs to be a framework that can seamlessly adapt analytical jobs located on a host site for execution on the cloud, provide minimal difference between a personal desktop and the cloud, and offer data and resource management easily on the cloud.

In this paper, such a framework, ‘RBioCloud’, which is light-weight and easily deployable has been designed and developed to support analytical jobs comprising R scripts which employ Bioconductor packages. The framework is deployed between a host site and the cloud, and a set of five command line tools are offered for analytical workflows to facilitate gathering resources, submitting a job, executing a job, retrieving results, and terminating resources. The research contributions of RBioCloud has been a framework to (i) seamlessly handle a diverse range of analytical job on the cloud, (ii) abstract the complexities of cloud set up and configuration, (iii) easily access and manage cloud resources, thereby saving time of domain scientists, and (iv) remotely access cloud resources from a workstation with seemingly minimal differences. Test cases using Bioconductor and R-based jobs demonstrate the feasibility of RBioCloud. Two test cases have been employed to validate the feasibility of RBioCloud. In the first test case, genome searching was performed on a single Amazon EC2 instance. In the second test case, normalisation of microRNA (miRNA) microarray data was performed using a cluster of Amazon EC2 instances. The framework is available for download from http://www.rbiocloud.com.

REFERENCES