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

Blesson Varghese, *Member, IEEE,* Ishan Patel, and Adam Barker, *Member, IEEE*

Abstract—Large-scale ad hoc analytics of genomic data is popular using the R-programming language supported by over 700 software packages provided by Bioconductor. More recently, analytical jobs are benefitting 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. Three biological test cases validate the feasibility of RBioCloud. The framework is available from `http://www.rbiocloud.com`.

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

1 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 is popular and is supported through over 700 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. Not only is analytics inherently computationally intensive, but also data intensive. High-performance computing systems have therefore become attractive for executing large-scale analytical jobs [3].

Traditional high-performance computing systems such as clusters and supercomputers offer a good platform to perform large-scale analytics. However, it is required of the computational biologist and bioinformaticist, who has excellent programming and statistical skills, to also have extensive knowledge of the high-performance computing hardware. Moreover, the costs required for investing in large-scale systems and their maintenance is high. The cloud has become an appealing alternative high-performance computing platform for ad-hoc analytics since it offers on-demand computing and storage resources, along with scalability and low maintenance costs [4], [5], [6], [7]. This has led to a variety of research for supporting analytics in computational biology and bioinformatics on the cloud (for example, [8], [9], [10] and [11]).

Software projects such as those reported in [12], [13] and [14] 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 by such projects for a fully configurable cloud cluster can fit well with the skill set of a cloud developer, thereby narrowing their wide usage. The major challenge in the research of developing software similar to the ones above for Computational Biology and Bioinformatics (for example, [15], [16], [17], [18], [19] and [20]) is to seamlessly execute an analytical job on the cloud in a manner similar to how the job would be executed on the personal workstation. However, the use of such software adds an additional layer of complexity for managing software on top of executing the job. Further, adapting the above 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 [21] on the cloud.

Every job needs to be manually configured on the cloud dashboard in the existing solutions using the Bioconductor cloud image[22]. The user needs to toggle between multiple screens and SSH into a remote machine. This is challenging and not suitable for executing multiple ad hoc tasks quickly. There needs to be simpler and automated ways of accessing the cloud for workloads. For example, a drop box like mechanism where the user can work on a remote machine with seemingly little difference to his own workstation and at the same time execute a job without configuring the remote machine for

each job. Efforts to automate the access of clouds, for example, [23], [24] and [25], are restricted to specific applications. These challenges can be overcome by the development of a generic framework that facilitates the 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. 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 contributions of the RBioCloud research are: (i) developing a light-weight framework for supporting a diverse range of analytical workloads using R and Bioconductor on the cloud, (ii) minimising human intervention for executing workloads on the cloud, (iii) abstracting complexities of configuring and executing jobs on the cloud, (iv) providing easy access and use of the cloud for computational biologists, and (v) executing an analytical workload on the cloud with seemingly minimal difference between a domain scientist's workstation and remote resources in the cloud.

The feasibility of RBioCloud is validated using three test cases employing Bioconductor packages for executing R-based scripts on the cloud. In the first test case, genome searching is performed on a single cloud instance, in the second test case, differentially expressed genes are detected on a single cloud instance, and in the third test case, normalisation of microRNA (miRNA) microarray data is performed on a cluster in the cloud.

The remainder of this paper is organised as follows. Section 2 considers the design of the RBioCloud framework. Section 3 describes the command line tools offered by RBioCloud for managing and executing an analytical job. Section 4 presents three test cases to validate the feasibility of RBioCloud. Section 5 concludes this paper by considering future work.

2 Framework Design

Figure 1, shows the design of the RBioCloud framework, which is sandwiched between a host site and the cloud. 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 2):

- *Step 1:* Gather resources - initialise cloud compute and storage resources from the host.

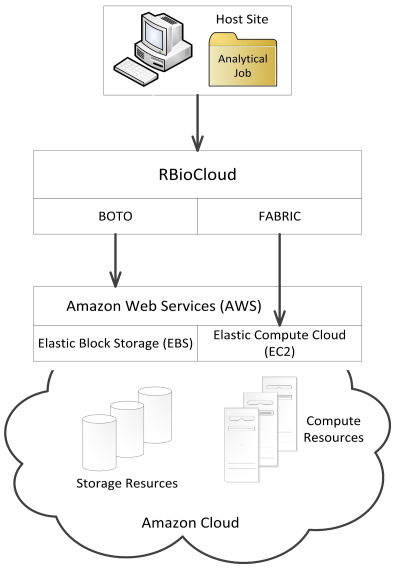


Fig. 1: Design of RBioCloud framework

- *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.

2.1 Supporting Interfaces

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)[1]. 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)[2] and are available as instances. The storage resources are referred to as the Elastic Block Storage (EBS)[3] that provide persistent data storage. Two Python interfaces, namely BOTO[4] provides the interface to access the resources provided by AWS and Fabric[5] facilitates remote administration of the cloud resources.

Amazon instances are initialized using Amazon Machine Images (AMI)[6]. The RBioCloud framework is built on the Bioconductor Cloud AMI [22] 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

1. http://aws.amazon.com/
2. http://aws.amazon.com/ec2/
3. http://aws.amazon.com/ebs/
4. https://github.com/boto/boto
5. http://docs.fabfile.org/en/1.4.3/
6. http://aws.amazon.com/amis

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2014.2361327, IEEE/ACM Transactions on Computational Biology and Bioinformatics

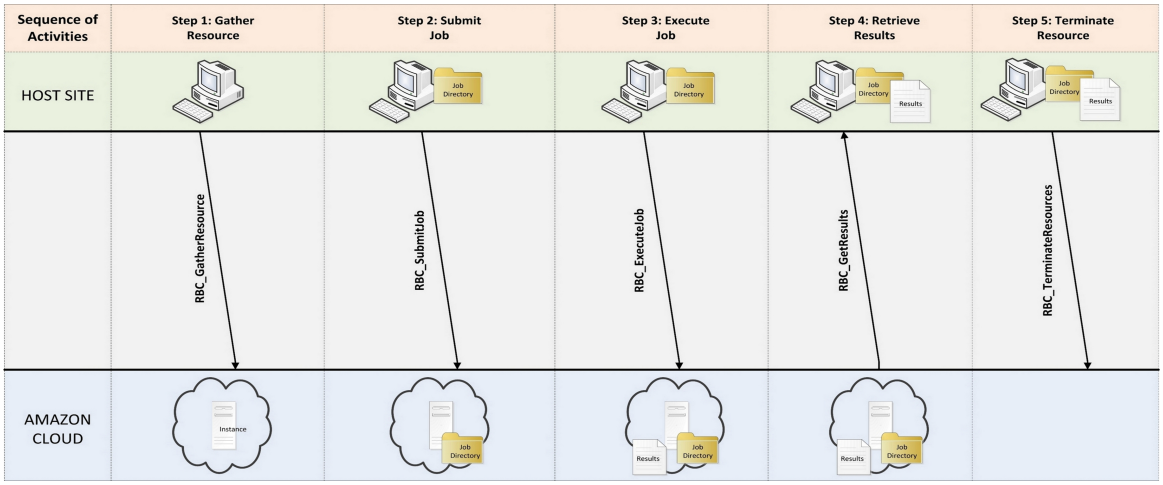


Fig. 2: Sequence of activities while using the RBioCloud framework

exploited on the cloud. The Simple Network Of Workstations (SNOW)[7] interface is employed for parallel execution of jobs on the cloud.

3 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.

3.1 Gather Resources

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

```
RBC_GatherResource [-h] [-v] [-rname
RESOURCE_NAME] [-rsize RESOURCE_SIZE]
[-ebsvol EBS_VOLUME | -snap EBS_SNAP]
[-type INSTANCE_TYPE] [-desc
RESOURCE_DESCRIPTION]
```

The optional arguments are: (a) `rname` to name a resource (instance or cluster) that is created, (b) `rsize`, to specify the size of the resource (if size is one, then only one instance is created, else if size is greater than one, then the instances are configured as a cluster), (c) `ebsvol` and `snap`, which are not specified at the same time. `ebsvol` specifies the EBS volume ID when an EBS volume is created. `snap` specifies the EBS snapshot ID from which an EBS volume can be created. `ebsvol` can be specified when an EBS volume is available, however, if `snap` is specified then a new EBS volume is created from the snapshot specified. If both arguments are not provided, then a default snapshot from a configuration file is used, (d) `type`, which defines the Amazon EC2 instance type[8] based on the computational requirements of the task, and (e) `desc`, which can be used to provide a description for a resource.

7. http://www.sfu.ca/~sblay/R/snow.html
8. http://aws.amazon.com/ec2/instance-types/

3.2 Submit Job

A job comprises the script that needs to be executed and the data required by the script both of which need to be submitted to the cloud. The 'rsync' protocol is used to submit the job. 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]
```

The optional arguments are: (a) `rname` to specify the resource to which the job needs to be submitted. If a resource is not specified then the default resource from RBioCloud's configuration file is employed, (b) `jobdir` to specify the job directory at the host. If the job directory is not specified then the current working directory at the host site is considered to be the source job directory. The destination job directory is not provided since in the current setup the host job directory is synchronised to the home directory of the root user on the cloud. The job directory comprises a set of R scripts, a set of data files required by the scripts and a sub-directory that will contain results after the execution of the script.

The optional switch `-tomaster` (default) submits the job to the master node of a cluster, while `-toallnodes` submits the job to all nodes of a cluster. The `-data` switch synchronises any folder not adhering to the structure of the job directory on to the resource.

3.3 Execute Job

`RBC_ExecuteJob` executes a job on the cloud resource. This command locks the resource onto the job and is only available for any additional use after the job has completed. The syntax of the command is

```
RBC_ExecuteJob [-h] [-v] [-rname
RESOURCE_NAME] [-jobdir JOB_DIRECTORY]
[-rscript R_SCRIPT] [-runname
RUN_NAME]
```

The optional arguments are: (a) `rname` to specify the resource on which the job needs to be executed, (b) `jobdir` to indicate the job directory at the host site; the job with the same name from the corresponding job directory on the cloud is executed, and (d) `rscript` to indicate the R script to be executed. If `rscript` is not provided then the user is prompted to select from a list of R scripts that are available in the job directory.

The mandatory argument `runname` provides a name each time a job is executed to distinguish multiple executions of the job.

3.4 Retrieve Results

`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]
```

The optional arguments are: (a) `rname` to specify the resource from where the results need to be retrieved, (b) `jobdir` to indicate the location of the source job directory at the host site; the results are fetched from the corresponding job directory on the cloud. If no job directory is specified then the current working directory at the host site is used.

The mandatory argument `runname` indicates the name of the specific execution whose results need to be gathered. This argument can be used when the same R script has been executed a number of times and each execution had to be differentiated. Within the job directory the results are generated in a sub-directory. There are two scenarios of generating results on a cluster. In the first scenario, the master instance aggregates and stores results from all worker instances, and retrieval from the master instance is possible using `-frommaster`. In the second scenario, the results are generated on all instances and are retrieved using `-fromall`.

3.5 Terminate Resources

After the completion of a job, the resources on the cloud need to be safely released to avoid billing of unused resources. `RBC_TerminateResource` facilitates this and the syntax is

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

The optional arguments are: (a) `rname` to specify the resource that needs to be terminated. The optional switch `-deletevol` deletes the EBS volume attached to the resource being terminated.

All the above commands can be used with two switches; firstly, `-h` to provide a description of the use and arguments of the command, and secondly, `-v` to provide the version of the installation.

4 Feasibility Study

In this section three test cases, firstly, genome searching, secondly, detecting differential expression of genes and thirdly, normalisation of microRNA (miRNA) microarray data are presented to demonstrate the feasibility of RBioCloud for Bioconductor and R based jobs. The software packages for the test cases are available from Bioconductor[9]. In the first and second test cases a single Amazon EC2 instance is used while in the third test case a cluster of Amazon EC2 instances are employed. The screenshots of the execution of the test cases are available at http://rbiocloud.com/download/1.0/RBioCloud-TestCases-1.0.pdf.

4.1 Test case 1: Genome searching on an Instance

The first test case is based on the `BSgenome` software package [26] 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 [27]. 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 are 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'
```

9. http://www.bioconductor.org/packages/release/bioc/

```
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 the 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 using two methods and produce their result in two separate files. The input is a dictionary, containing 50 patterns, each of which is a short nucleotide sequence of 15 to 25 bases. In the first method, the forward and reverse strands of seven Caenorhabditis elegans chromosomes named as chrI, chrII, chrIII, chrIV, chrV, chrX, chrM are the target. The result obtained is in a tabulated form in `ce2dict0_ana1.txt` providing the name of the chromosome where the hit occurs, two integers giving the starting and ending positions of the hit, an indication of the hit either in the forward or reverse strand, and unique identification for every pattern in the dictionary. A sample of the output in `ce2dict0_ana1.txt` is shown in Figure 3 (left).

In the second method, a function which is approximately one hundred times faster is employed. One limitation of the function is that it works only when all DNA patterns searched for have a constant number of nucleotide bases. Therefore, the nucleotide patterns in the dictionary are truncated to a constant length of 15. The output of this method is also tabulated in the second result file `ce2dict0cw15_ana2.txt` in a similar way to the first method. A sample of the output is shown in Figure 3 (right).

4.2 Test case 2: Detection of differentially expressed genes on an Instance

The second test case is based on the `logitT` software package [28]. The script executed is `logitT.R` which is a statistical method based on the Logit-t algorithm for identifying differentially expressed genes using probe-level data. The input to the script is the `spikein95` data set of the `SpikeInSubset` library [29]. This data set is a subset of the Human Genome U95 data set containing a series of genes spiked-in at known concentrations and arrayed in a Latin Square format[10]. The Logit-t algorithm requires limited pre-processing before the actual statistical analysis and produces better results [30] compared to competing approaches such as the regression modelling approach [31] and the Significance Analysis of Microarrays (SAM) [32].

For executing the script using RBioCloud, the job is organised into one directory, for example `logitT`, which contains the `logitT.R` script, all associated data and the `Results` and `RunResults` directories. The following sequence of five commands will execute `logitT.R` on the cloud and fetch the results onto the host site:

```
1 > RBC_GatherResource -rname
    'logitT_instance' -rsize 1 -desc
    'For_Detecting_Differentially_Express-
    ed_Genes'
2 > RBC_SubmitJob -rname
    'logitT_instance'
3 > RBC_ExecuteJob -rname
    'logitT_instance' -rscript
    'logitT.R' -runname
    'Run1_on_logitT_instance'
4 > RBC_GetResults -rname
    'logitT_instance' -runname
    'Run1_on_logitT_instance'
5 > RBC_TerminateResource -rname
    'logitT_instance' -deletevol
```

When the `logitT.R` script is executed on the `logitT_instance`, firstly, probe level intensities are normalised using the logit-log transformation. Then the probe level intensities are standardised using Z-transformation. Student's t-tests are performed for every Perfect Match (PM) probe in a probe set. The median t-statistic for the probe set defines Logit-t. The p-values of all the probe sets are calculated and probe sets with p-values less than 0.01 marks the detection of differentially expressed genes. The output of the algorithm is as follows:

```
"1024_at"  "1708_at"  "32660_at" "36202_at"
"36311_at" "38734_at"
```

10. http://www.affymetrix.com/support/technical/sample_data/datasets.affx

```
seqname start    end      strand  patternID
chrI    5942496  5942511  -       pattern17
chrI    6298363  6298377  +       pattern19
chrI    12760564 12760587 -       pattern21
chrI    3953136  3953150  +       pattern23
chrI    11568996 11569018 -       pattern27
chrI    753618   753641   +       pattern37
...
```

```
seqname start   end     strand  patternID
...
chrI    13745040        13745054        +  pattern04
chrI    14075187        14075201        +  pattern04
chrI    11745177        11745191        +  pattern08
chrI    8981081 8981095 +       pattern11
chrI    12188778        12188792        +  pattern16
chrI    12233665        12233679        +  pattern16
...
```

Fig. 3: Sample results obtained from first method (left) and second method (right) in `GenomeSearching.R`

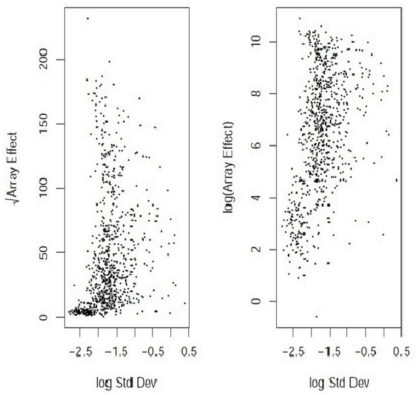
4.3 Test case 3: Normalisation of microRNA (miRNA) microarray data on a Cluster

The third test case is based on the `LVSmiRNA` software package [33] and the script executed is `LVSmiRNA.R` which normalises microRNA (miRNA) microarray data. The Least-Variant Set (LVS) normalisation method [34] is employed in the package and the input is the miRNA expression data [35] 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 4a) with logarithmic scales showing the array effect versus standard deviation. The second result obtained from the script is a box plot (refer Figure 4b) of data after normalisation.

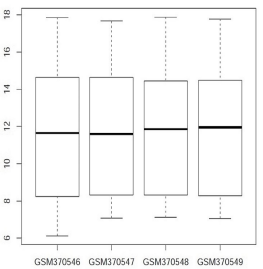
The `estVC` function can benefit from using parallel computation for achieving higher speed up over sequential computation, and can take a cluster object as an argument. Here Amazon clusters can come to play, and will need to be manually configured using the Amazon dashboard as shown in [22] and [36]. 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) Scatter plot on logarithmic scale showing array effect versus standard deviation obtained from `LVSmiRNA.R`



(b) Box plot of miRNA data after LVS normalisation obtained from `LVSmiRNA.R`

Fig. 4: Results from Test case 3

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 is retrieved on to the host `Results` directory when the fourth command is executed. The Amazon resource `LVSmiRNA_cluster` is terminated using the fifth command.

4.4 Summary

Figure 5 is a graph showing the time taken to move data related to the job in and out of the cloud. The Amazon resources used for test case 1 and test case 2 are one m1.xlarge instance and for test case 3 is a cluster of six m1.xlarge instances. There is an increase in the time taken for initialising and terminating the cluster over the time taken for initialising and terminating one instance. Therefore, alternative techniques will need to be considered for initialising and terminating resources in parallel. This can contribute to the reduction of the overall time taken by RBioCloud.

The time taken to submit the job is proportional to the size of the script and the input data being submitted. Large data sets required by the three test cases are available on the Amazon instances employed in this research (a custom built Amazon Machine Image (AMI) based on the Bioconductor AMI is used in this research). Genome searching takes 79 seconds and the detection of differential expression of genes takes 41 seconds to complete execution. The potential for parallelism in these jobs and scaling the job across multiple instances need to be explored to achieve speed up. The third test case exploits parallelism and executes on a cluster of six instances taking 18 seconds for completing the job. Again the time for retrieving results is proportional to the size of the files produced as results. The second test case takes the least time for retrieval since it produces a small output.

Additional test cases to confirm the feasibility of RBioCloud were performed on over 150 Bioconductor scripts. A discussion of the results obtained from the additional test cases are beyond the scope of this paper; the results can be obtained from http://www.rbiocloud.com/download/1.0/RBioCloud-Datasheet-1.0.pdf. One observation from the test cases is that the full advantage of the cloud is exploited when jobs harness the potential of parallelism.

5 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 between a host site that has the analytical job and the cloud that can seamlessly adapt analytical jobs 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 (i) gathering resources, (ii) submitting a job, (iii) executing a job, (iv) retrieving results, and (v) terminating resources. Test cases using Bioconductor and R-based jobs demonstrate the feasibility of RBioCloud. Three test cases have been employed to validate the feasibility of RBioCloud. In the first test case, genome searching, and in the second test case, detection of differential expression of genes were both 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. Results from additional test cases and the framework are available for download from `http://www.rbiocloud.com`.

Future efforts will be made towards extending RBioCloud for multiple public cloud providers such as Microsoft Azure and Rackspace. Additonally, the framework will be extended to accommodate parallelism without explicit programming. For this, a job will need to be split into parallel components that can be executed on the virtual cores of the cloud instances. Parallelism can be fully taken advantage of if these mechanisms can be extended to support hardware accelerators, such as GPUs, in RBioCloud.

References

[1] R Core Team (2012) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Website: http://www.R-project.org.

[2] R. Gentleman, V. Carey, D. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Yang and J. Zhang, "Bioconductor: Open Software Development for Computational Biology and Bioinformatics," Genome Biology 5(10): R80, 2004.

[3] E. E. Schadt, M. D. Linderman, J. Sorenson, L. Lee and G. P. Nolan GP, "Computational Solutions to Large-Scale Data Management and Analysis," Nature Reviews Genetics 11(9), 2010.

[4] J. T. Dudley, Y. Pouliot, R. Chen, A. A. Morgan and A. J. Butte, "Translational Bioinformatics in the Cloud: An Affordable Alternative," Genome Medicine 2(8), 2010.

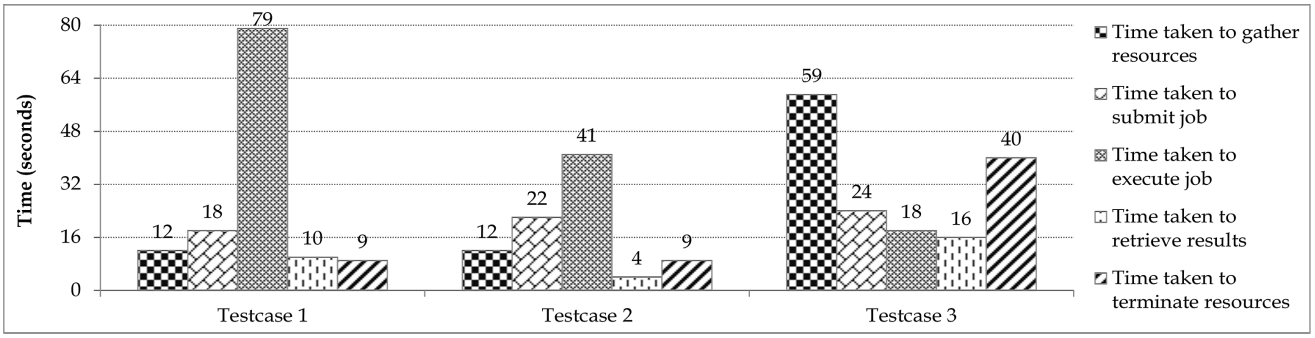


Fig. 5: Time taken by the test cases on RBioCloud for gathering resources, submitting and executing jobs, retrieving results and terminating resources

[5] S. V. Angiuoli, J. R. White, M. Matalka, O. White and W. F. Fricke, "Resources and Costs for Microbial Sequence Analysis Evaluated Using Virtual Machines and Cloud Computing," PLoS ONE 6(10), 2011.

[6] D. P. Wall, P. Kudtarkar, V. A. Fusaro, R. Pivovarov, P. Patil and P. J. Tonellato, "Cloud Computing for Comparative Genomics," BMC Bioinformatics 11(1), 2010.

[7] M. C. Schatz, B. Langmead and S. L. Salzberg, "Cloud Computing and the DNA Data Race," Nature Biotechnology 28: 691-693, 2010.

[8] L. D. Stein, "The Case for Cloud Computing in Genome Informatics," Genome Biology 11(5), 2010.

[9] F. N. Memon, A. M. Owen, O. Sanchez-Graillet, G. J. Upton and A. P. Harrison, "Identifying the Impact of G-Quadruplexes on Affymetrix 3' Arrays Using Cloud Computing," Journal of Integrative Bioinformatics 7(2), 2010.

[10] B. D. Hallgan, J. F. Geiger, A. K. Vallejos, A. S. Greene and S. N. Twigger, "Low Cost, Scalable Proteomics Data Analysis Using Amazon's Cloud Computing Services and Open Source Search Algorithms," Journal of Proteome Research 8(6): 3148-3153, 2009.

[11] V. A. Fusaro, P. Patil, E. Gafni, D. P. Wall and P. J. Tonellato, "Biomedical Cloud Computing with Amazon Web Services," PLoS Computational Biology 7(8), 2011.

[12] K. Chine, "Scientific Computing Environments in the Age of Virtualization, Toward a Universal Platform for the Cloud," Proceedings of the IEEE International Workshop on Open-source Software for Scientific Computation, 44-48, 2009.

[13] J. Kim, S. Maddineni and S. Jha, "Building Gateways for Life-Science Applications using the Dynamic Application Runtime Environment (DARE) Framework," Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery, 2011.

[14] W. Lu, J. Jackson and R. Barga, "AzureBlast: A Case Study of Developing Science Applications on the Cloud," Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, 413-420, 2010.

[15] A. Matsunaga, M. Tsugawa and J. Fortes J, "CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications," Proceedings of the 4th IEEE International Conference on eScience, 222-229, 2008.

[16] E. Afgan, D. Baker, N. Coraor, B. Chapman, A. Nekrutenko and J. Taylor, "Galaxy CloudMan: Delivering Cloud Compute Clusters," BMC Bioinformatics 11(Suppl 12), 2010.

[17] M. Fischer, R. Snajder, S. Pabinger, A. Dander, A. Schossig, J. Zschocke, Z. Tranjanoski and G. Stocker, "SIMPLEX: Cloud-Enable Pipeline for the Comprehensive Analysis of Exome Sequencing Data," PLoS ONE 7(8), 2012.

[18] B. Langmead, M. C. Schatz, J. Lin, M. Pop and S. L. Salzberg, "Searching for SNPs with Cloud Computing,' Genome Biology 10(11), 2009.

[19] M. C. Schatz, "CloudBurst: Highly Sensitive Read Mapping with MapReduce," Bioinformatics 25(11): 1363-1369, 2009.

[20] S. Angiuoli, M. Matalka, A. Gussman, K. Galens, M. Vangla, D. R. Riley, C. Arze, J. R. White, O. White and W. F. Fricke, "CloVR: A Virtual Machine for Automated and Portable Sequence Analysis from the Desktop Using Cloud Computing," BMC Bioinformatics 12(1), 2011.

[21] R. Gentleman, V. Carey, W. Huber, R. Irizarry and S. Dudoit, "Bioinformatics and Computational Biology Solutions Using R and Bioconductor," Springer, 494 p, 2005.

[22] Bioconductor in the cloud website: http://www.bioconductor.org/help/bioconductor-cloud-ami/

[23] B. Langmead, K. D. Hansen and J. T. Leek, "Cloud-Scale RNA-Sequencing Differential Expression Analysis with Myrna," Genome Biology 11(8), 2010.

[24] Contrail website: http://sourceforge.net/apps/mediawiki/contrail-bio/index.php?title=Contrail

[25] Jnomics website: http://sourceforge.net/apps/mediawiki/jnomics/index.php?title=Jnomics

[26] H. Pages, "BSgenome: Infrastructure for Biostrings-based Genome Data Packages," R package version 1.26.1, 2012.

[27] A. G. Fraser, R. S. Kamath, P. Zipperlen, M. Martinez-Campos, M. Sohrmann and J. Ahringer, "Functional Genomic Analysis of C. elegans Chromosome I by Systematic RNA Interference," Nature 408: 325-330, 2000.

[28] T. Guennel, "logitT: logit-t Package," R package version 1.18.0, 2008.

[29] R. Irizarry and Z. Wu, "SpikeInSubset: Part of Affymetrix's Spike-In Experiment Data," R package version 1.2.13, 2013.

[30] W. J. Lemon, S. Liyanarachchi and M. You, "A High Performance Test of Differential Gene Expression for Oligonucleotide Arrays," Genome Biology 4: R67, 2003.

[31] J. G. Thomas, J. M. Olson, S. J. Tapscott and L. P. Zhao, "An Efficient and Robust Statistical Modelling Approach to Discover Differentially Expressed Genes Using Genomic Expression Profiles," Genome Research 11(7): 1227-1236, 2001.

[32] V. G. Tusher, R. Tibshirani and G. Chu, "Significance Analysis of Microarrays Applied to the Ionizing Radiation Response," Proceedings of the National Academy of Science, 98, 5116-5121, 2001.

[33] S. Calza, S. Chen and Y. Pawitam, "LVSmiRNA: LVS Normalization for Agilent miRNA Data," R package version 1.8.0, 2010.

[34] S. Calza, D. Valentini and Y. Pawitan, "Normalization of Oligonucleotide Arrays Based on the Least-Variant Set of Genes," BMC Bionformatics 140: 5-9, 2007.

[35] H. Willenbrock, J. Salomon, R. Sokilde, K. B. Barken, T. N. Hansen, F. C. Nielsen, S. Moller and T. Litman, "Quanitative miRNA Expression Analysis: Comparing Microarrays with Next-Generation Sequencing," RNA 15(11): 2028-2034, 2009.

[36] Amazon Elastic Compute Cloud, "Getting Started Guide," API Version 2012-10-01, 2012.

[37] B. Efron, R. Tibshirani, J. D. Storey, and V. Tusher, "Empirical Bayes Analysis of a Microarray Experiment," Journal of the American Statistical Association, 96, 1151-1160, 2001.

[38] H. Schwender, "siggenes: Multiple testing using SAM and Efron's Empirical Bayes approaches," R package version 1.34.0, 2012.

[39] J. M. Muino, K. Kaufmann, R. C. H. J. van Ham, G. C. Angenent and P. Krajewski, "ChIP-seq Analysis in R (CSAR): An R Package for the Statistical Detection of Protein-Bound Genomic Regions, Plant Methods, 7:11, 2011.

Blesson Varghese is a Research Fellow at the School of Computer Science, University of St Andrews, UK. Prior to taking up this position he was a post-doctoral researcher at the Dalhousie University, Canada, where he pursued the work on RBioCloud. He obtained his PhD from the University of Reading, UK, on multiple international scholarships. His research is currently supported by an EPSRC Impact Acceleration Account Grant and three industry awards from Amazon Web Services, Microsoft Azure and NVIDIA. E-mail: varghese@st-andrews.ac.uk, Web: http://www.blessonv.com.

Ishan Patel works with IBM Canada Ltd., Canada. Prior to taking up this position he completed his Master's at the Dalhousie University, Canada, where he worked on RBioCloud. He has experience in developing cloud-based tools required in scientific and engineering domains. E-mail: ishanp@ca.ibm.com

Adam Barker is a Senior Lecturer at the School of Computer Science, University of St Andrews, UK. He is currently a Royal Society Industry Fellow working with Cloud-Soft Corporation Ltd, Edinburgh, UK, and an Honorary Fellow at the University of Edinburgh, UK. Prior to obtaining a faculty position he worked as a Research Fellow at the University of Oxford, UK, the University of Melbourne, Australia, and the University of Edinburgh, UK. He obtained his PhD from the University of Edinburgh, UK. His research is supported by EPSRC, Royal Society, Royal Society of Edinburgh, Impact Acceleration Account, SICSA, Amazon and Microsoft. E-mail: adam.barker@st-andrews.a.uk, Web: http://www.adambarker.org.