A Platform for Parallel R-based Analytics on Cloud Infrastructure

Ishan Patel, Andrew Rau-Chaplin and Blesson Varghese[1]
Risk Analytics Lab, Faculty of Computer Science, Dalhousie University, Halifax, Canada
Email: {patel, arc, varghese}@cs.dal.ca

***Abstract*—Analytical workloads abound in application domains ranging from computational finance and risk analytics to engineering and manufacturing settings. In this paper we describe a Platform for Parallel R-based Analytics on the Cloud (P2RAC). The goal of this platform is to allow an Analyst to take a simulation or optimization job (both the code and associated data) that runs on their personal workstations and with minimum effort have them run on large-scale parallel cloud infrastructure. If this can be facilitated gracefully, an Analyst with strong quantitative but perhaps more limited development skills can harness the computational power of the cloud to solve larger analytically problems in less time. P2RAC is currently designed for executing parallel R scripts on the Amazon Elastic Computing Cloud infrastructure. Preliminary results obtained from an experiment confirm the feasibility of the platform.**

***Index Terms*—Cloud computing; Amazon Cloud Services; Parallel Analytics; R and Snow**

I. Introduction

Cloud based infrastructure has proven to be very effective in providing on-demand computational resources to both commercial applications and a wide range of large-scale scientific applications (e.g. climate simulation and analysis [1], biomedical image processing [2], fMRI brain imaging [3], satellite ground systems [4] and data processing [5], astronomy applications [6], Geographic Information Systems (GIS) [7] and disaster response systems [8]). In both the commercial and scientific settings the talents of computer scientists and expert developers are brought to bear in order to efficiently exploit cloud-based infrastructure.

In this paper, we explore how a different class of users with a different kind of workload might be able to take advantage of the cloud. In particular, we study how Analysts, who are domain experts with quantitative/mathematical skills, but often with software skills limited to high-level programming environments like R [9], Matlab [10] or Octave [10], might be supported in harnessing the cloud for ad hoc analytical workloads.

Analytical workloads abound in application domains ranging from computational finance and risk analytics to engineering and manufacturing settings. In our experience, these workloads which involve typically involve simulation [12] and optimization [11] tasks share some common features, namely

(a) The associated codes are developed by Analysts (not professional developers) in high-level programming environments such as R or Matlab.

(b) These codes and the related input data are generally created by Analysts for either one time use or are heavily modified each time they are used to adapt them to the analytical question at hand.

(c) The codes are often computationally intensive or require a large number of independent runs with varying input parameters making some form of parallelism attractive.

Our goal has been to develop a platform that allows an Analyst to take an analytical job (both the code and associated data) that runs on their personal workstations and with minimum effort have them run on large-scale parallel cloud infrastructure. If this can be facilitated gracefully, the Analyst can solve larger problems or perform more experiments in less time. Our approach is somewhat different from other 'cluster on cloud' projects such as [13][14][15][16] in that our focus is to simplify an Analyst's use of cloud infrastructure, rather than provide a fully-configurable high-performance computing cluster on clouds for developers. In particular, we have explored a platform for facilitating R-based risk analytics on the Amazon cloud. However, we believe that the basic platform and the experienced gained can be generalized to a wider class of analytics and cloud-based infrastructure.

The remainder of this paper is organized as follows. Section II presents the design of the **P**latform for **P**arallel **R**-based **A**nalytics on **C**loud Infrastructure (P2RAC). Section III considers the implementation of P2RAC and highlights the key tools offered by the P2RAC platform. A sample workflow presenting the sequence of commands to run an Analyst project and preliminary experimental results obtained from PR2RAC are considered in Section IV. The paper concludes and future work is discussed in Section VI.

II. The Platform

In this section, the Platform for Parallel R-based Analytics on the Cloud, referred to as P2RAC, is presented. Two considerations are taken into account, firstly, the conceptual design of the platform, and secondly, how the platform fits in coherently between an Analyst and the cloud infrastructure. See Figure 1.

In this paper, an Analyst is considered as the typical user who can make use of the cloud infrastructure. The Analyst needs to perform a large-scale analysis of data, for example, analytics for optimization problems in the (re)insurance industry using the R programming language. Owing to the

[1]Corresponding Author

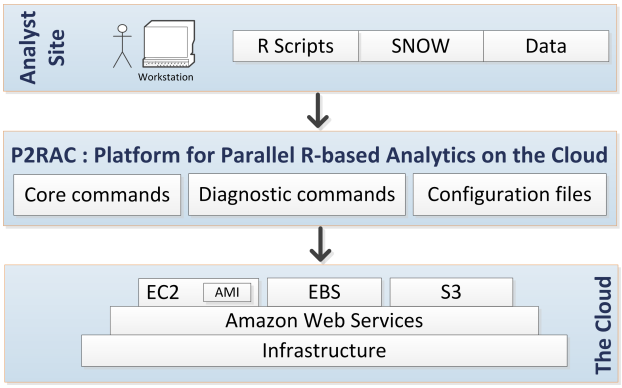

Fig. 1: Conceptual design of P2RAC

large-scale nature of the size of data and its analysis, parallel computing [17] is therefore a paradigm that is employed. The Simple Network Of Workstations (SNOW) package facilitates parallel computations in R [18]. An Analyst may use a project comprising a number of R scripts (using SNOW for parallel computations) and large data files.

The cloud infrastructure that is intended to be exploited is a pool of computational and storage resources geographically dispersed. The access to the resources provided by a cloud is facilitated by a service, namely the Infrastructure as a Service (IaaS) [19]. Amazon is a leading provider of IaaS and their web services, including the Elastic Compute Cloud (EC2) [20] and the Elastic Block Storage (EBS) [21] are employed by the platform developed in the research reported in this paper.

The P2RAC platform mediates between an Analysts project and the web services offered by Amazon. The platform is built with the Python programming language and draws heavily on two libraries. Firstly, the Boto library, which provides P2RAC a Python interface to the Amazon web services. Secondly, the Fabric library, which facilitates remote administration of the cloud nodes for P2RAC. The platform comprises three components, (a) the core tools, (b) the diagnostic tools and (c) the configuration files. The core tools provide functionalities for cluster management, data management and execution management. The diagnostic tools provide functionalities for checking the computational environment. The configuration files provide support for the functionalities of the core and diagnostic tools.

The workflow of how an Analyst could potentially harness the cloud computing services offered by Amazon is shown in Figure 1. The Analyst who has a project comprising R scripts using SNOW or other parallel libraries and large data files provides the project as a task for execution on the cloud to the P2RAC platform. Then the platform both gathers and initializes a pool of computational and storage resources on the cloud for executing the task. The platform subsequently transfers the task onto the pool of resources and manages task execution. After the task is executed, the platform facilitates the gathering of results which may be spread across the pool of resources in the cloud. Finally, the pool of resources is released.

The next section considers the working of the core tools, the diagnostic tools and the configuration files and how they are implemented in the P2RAC platform.

III. Implementation

As considered in the above section, the P2RAC platform comprises three components, namely the core tools, the diagnostic tools and the configuration files. The core tools and the diagnostic tools are implemented as commands which can be executed from the command-line.

A. Core Tools

The core tools of P2RAC provide the functionalities for cluster management, data management and execution management of a task on the Amazon cloud. The core tools provide the Analyst with an interface to the cloud which allows significant computational resources to be brought to bear while greatly reducing the complexities associated with directly working with the cloud infrastructure. This is necessary since an Analyst is less likely to have knowledge and experience of working with computational clouds.

1) Cluster Management: Cluster management in the core tools are a set of functionalities that range from gathering a pool of resources on the cloud, followed by configuring the pool of resources as a cluster, offering the cluster for task execution and finally terminating the cluster when the task executing on the cluster has completed. P2RAC offers two core tools for cluster management, namely `ec2createcluster` and `ec2terminatecluster`.

The ec2createcluster tool is responsible for gathering and configuring the pool of resources as a cluster on the cloud. The computational resources on Amazon are referred to as Elastic Compute Cloud (EC2) and are available as instances. These resources are available on-demand and are paid for on the basis of their usage. The instances are initialized using Amazon Machine Images (AMI) [22]. For example, an AMI referred to as the Bioconductor Cloud AMI [23], is used in the research reported in this paper.

Similarly, storage resources on Amazon, such as the Elastic Block Storage (EBS) are available on-demand and are paid for on the basis of data transfer and volume of storage. One note worthy feature of EBS includes its ability to provide persistent data storage. This feature can be exploited when large volumes of data need to be used in an Analysts project and thereby eliminates the need for frequent transfer of data that may not be changed over time. Another feature of EBS is that it can be attached (mounted) as a local storage onto an EC2 instance in addition to its storage. This feature eliminates the need for additionally making data locally available to the instance.

The syntax of the `ec2createcluster` command is

```
ec2createcluster -cname <CLUSTER_NAME> -csize
<CLUSTER_SIZE> -ebsvol <EBS_VOLUME> -type
<INSTANCE_TYPE> -desc <CLUSTER_DESCRIPTION>
```

The optional arguments of the `ec2createcluster` command are `cname`, `csize`, `ebsvol`, `type` and `desc`. `cname` specifies the

name of the cluster that is created. `csize` specifies the size of the cluster. `ebsvol` specifies the EBS volume ID which is provided by Amazon when an EBS volume is created. `type` defines the Amazon EC2 instance type which is specified based on the computational requirements of the task. For example, a High-memory Quadruple Extra Large Instance, offers 68.4 GB of memory, 26 EC2 compute units, 1690 GB of instance storage, with high input/output performance, with instance type as `m2.4xlarge`, and was employed in the work reported in this paper. The `desc` argument can be used to provide a description of the cluster.

For example, if a command such as

```
ec2createcluster -cname 'hpc_cluster' -csize '10'
-ebsvol 'vol-xxxxxxxx' -type 'm2.4xlarge' -desc 'For
Trial Simulation Run'
```

is executed, then a sequence of activities follow. Ten EC2 instances of `m2.4xlarge` type are initialized using the Bioconductor Cloud AMI (The AMI ID is provided in the configuration file). The cluster of the ten EC2 instances is referred to as `hpc_cluster`. One instance in the `hpc_cluster` is denoted as the master and tagged as `hpc_cluster_Master`, while the remaining nine instances are denoted as workers and are tagged as `hpc_cluster_Workers`. The EBS volume, `vol-xxxxxxxx` (volume ID is masked in this paper) is attached on to the master instance. Network File System (NFS) is employed to share the attached EBS volume among the nine worker instances. A configuration file at the Analyst site is updated with cluster information such as the public DNS names of the master and worker instances, size of the cluster, EBS volume ID and description of the cluster. Libraries which an Analyst needs to include can additionally be installed by specifying library packages in another configuration file. In the research reported in this paper, the SNOW package and the R version of GENetic Optimization Using Derivatives (rgenoud) package [24] are additionally installed since the Bioconductor Cloud AMI does not provide these packages. Should the optional arguments be not provided then the default values which are defined in a configuration file is chosen.

The multiple execution of the `ec2createcluster` command facilitates the creation of multiple clusters. Since an EBS volume can only be attached to the master instance of one cluster the need for multiple EBS volumes arises when multiple clusters are created. Should multiple EBS volumes require the same data then they need to snapshot from the same source located on Simple Storage Service (S3) offered by Amazon [25]. Multiple clusters cannot have the same name when the `ec2createcluster` command is executed more than once.

When a task has completed execution it is essential to safely release the resources which are utilized by the cluster. To this end, the `ec2terminate` cluster command is provided. The syntax of the `ec2terminatecluster` command is

```
ec2terminatecluster -cname <CLUSTER_NAME> -deletevol
```

The optional arguments of the `ec2terminatecluster` command are `cname` and `deletevol`. `cname` specifies the name of the cluster that needs to be terminated. The `deletevol` switch deletes an EBS volume attached to the cluster being terminated.

For example, if a command such as

```
ec2terminatecluster cname 'hpc_cluster'
```

is executed, and then a sequence of activities follows. Firstly, the EBS volume that has been shared with the worker instances through NFS is no more shared. Further to this, the worker instances are terminated such that they do not exist. The EBS volume `vol-xxxxxxxx` is detached from the master node and the master instance is terminated. The section containing the cluster information of `hpc_cluster` in the configuration file is removed. Should the `deletevol` switch be included in the command then the EBS volume, `vol-xxxxxxxx` is deleted.

2) Data management: Data management is required to transfer the task (both the scripts and data) from the Analyst site to the cluster on the cloud, and thereafter receive results to the Analyst site. The transfer of task may be to the entire pool of resources in the cluster or to a specific instance on the cluster. Two feasible routes are to use the Secure Copy (SCP) protocol or the rsync protocol. The rsync protocol is employed owing to the quicker synchronisation of data between a source and a destination site. Therefore, two commands, namely the `ec2rsyncme` and `ec2rsyncmetomaster` based on the rsync protocol are provided. In order to receive the results to the Analyst site the `ec2getresults` command is developed.

The `ec2rsyncme` command enables an Analyst's project to be synchronised with all instances of a cluster. This stands different to the data that is stored in an EBS volume mounted on the master instance and shared with the worker instances. In this research, large volumes of data which are less likely to change in a short course of time are stored on the EBS volume. On the other hand smaller chunks of data that frequently change are synchronised from an Analyst's site on to the local storage of the cluster instances.

The structure of a project at the Analyst's site is worthwhile to be noted. A directory comprising a set of R scripts which need to be executed, a set of data files required by the scripts and a sub-directory that will contain results after the execution of the script. The `ec2rsyncme` command synchronises the directory from the Analyst's site to all the instances of the cluster. In other words, every instance contains a project directory.

The syntax of the `ec2rsyncme` command is

```
ec2rsyncme -cname <CLUSTER_NAME> -projectdir
<PROJECT_DIRECTORY>
```

The optional arguments of the `ec2rsyncme` command are `cname` and `projectdir`. The `cname` argument specifies the name of the cluster whose instances will be synchronised with the project directory. The source project directory is specified as the argument `projectdir`. If the cluster name is not provided by the Analyst then the cluster name from the configuration file is employed. Should the project directory not be specified then the current working directory at the Analyst site is used as the source project directory. The

destination directory is not specified since the project directory is synchronised at the home directory of the root user.

Owing to the nature of the task to be executed, it may not be necessary that the project directory be provided to all the instances of a cluster. For example, consider a task in which the master instance receives data from the Analyst site and distributes it to the worker instances in the cluster. In such a case it would be inefficient to synchronise the source project directory with all the instances of the cluster, but would be sufficient for the master instance alone to have the project directory. To facilitate this, `ec2rsyncmetomaster` command is provided.

The syntax of the `ec2rsyncmetomaster` is

```
ec2rsyncmetomaster cname <CLUSTER_NAME> -projectdir
<PROJECT_DIRECTORY>
```

The optional arguments of `ec2rsyncmetomaster` are similar to that of `ec2rsyncme`.

Based on the nature of the R scripts that are executed there are three possible scenarios for generating results. It is assumed that the R scripts generate results in a sub-directory within the project directory. In the first scenario, the master instance aggregates the results from the worker instances and stores them at the master instance. In the second scenario, however, the results are only generated on the worker instances. In the third scenario, the results are generated on both the master and worker instances. In both the scenarios, the results need to be obtained at the Analyst site. Therefore, the `ec2getresults` command is provided. To address the first scenario, `ec2getresults` gathers results from the master instance and provides it at the Analyst site. To address the second scenario, `ec2getresults` gathers results from the worker instances. In the third scenario, `ec2getresults` gathers results from both the master and all the worker instances. The aggregated results are stored in a directory at the same hierarchical level of the project directory at the Analyst site.

The syntax of the `ec2getresults` is

```
ec2getresults cname <CLUSTER_NAME> -projectdir
<PROJECT_DIRECTORY> -runname <RUN_NAME> -frommaster
| -fromworkers | -fromall
```

The optional arguments are `cname`, `projectdir` and a switch. The `cname` argument specifies the name of the cluster from where the results have to be fetched. The `projectdir` specifies the location of the source project directory at the Analyst site. The command utilises the name of the project from the path of the source project directory to fetch data from the corresponding project directory on the cluster. If no project directory is specified then the path of the current working directory at the Analyst site is used. The switch specifies the instances from where the results need to be gathered. If `frommaster` is specified, then results are gathered as in the first scenario. If `fromworkers` is specified, then results are gathered as in the second scenario. If `fromall` is specified, then results are gathered as in the third scenario. If no switch is specified then the results are gathered as in the first scenario.

The mandatory argument for the `ec2getresults` command is `runname` which indicates the name of a run that was specified during execution and whose results need to be gathered. This argument is used if the same R script has been executed a number of times and each execution had to be differentiated.

3) Execution Management: Execution management assigns the Analyst task onto a cluster and further runs task on the cluster. For this, the `ec2runrscript` command is provided. The syntax of `ec2runscript` is

```
ec2runscript -cname <CLUSTER_NAME> -projectdir
<PROJECT_DIRECTORY> -rscript <R_SCRIPT> -runname
<RUN_NAME>
```

The optional arguments of `ec2runrscript` are `cname`, `projectdir` and `rscript`. The `cname` argument specifies the name of the cluster where the R script needs to be executed. The `projectdir` specifies the location of the source project directory at the Analyst site. The command utilises the name of the project from the path of the source project directory to execute an R script from the corresponding project directory on the cluster. `rscript` specifies the name of the R script to be executed from `projectdir`. If `rscript` is not provided then the user is prompted to select from a list of R scripts that may be available in the project directory.

The mandatory argument for `ec2runrscript` is `runname` which indicates the name of a run.

B. Diagnostic Tools

P2RAC currently offers two tools, namely the `ec2listclusters` and `ec2logintomaster`, which are used for listing the clusters created by the Analyst on the Amazon cloud and for accessing the master instance of a cluster respectively.

The syntax of `ec2listclusters` is

```
ec2listclusters -nameonly
```

The optional switch `nameonly` provides the names of the clusters on the cloud. If the switch is not provided then the list of the clusters along with the size of the cluster, public DNS name of all instances, volume ID of the EBS volume shared with the instances of the cluster and the description of the cluster.

The syntax of `ec2logintomaster` is

```
ec2logintomaster -cname <CLUSTER_NAME>
```

The optional argument `cname` specifies the name of the cluster whose master instance needs to be accessed. The connection to the master instance is facilitated through Secure Shell (SSH). If the name of the cluster is not provided then the master instance of the default cluster listed in the configuration file is used.

C. Configuration Files

There are three files that support the core and diagnostic tools which reside on the Analyst site. Firstly, a file that contains a list of variables that are required by the command line tools along with a number of directory paths and references to access keys for Amazon resources. Secondly, a file that contains information, such as the names of the clusters created,

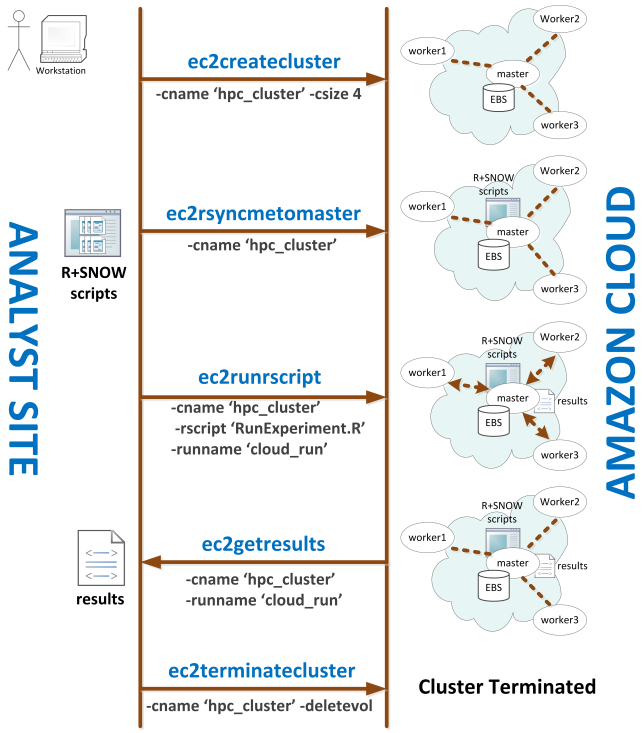

Fig. 2: Workflow of commands using P2RAC

their size, the public DNS names of all their instances, volume ID of the EBS volume shared by the master with the workers and the description of the clusters. Thirdly, a file that contains a list of R libraries which are required by an Analysts project. These libraries are installed on the instances of the cluster when it is created. This is required in addition to the pre-installed libraries of the base AMI.

All the commands considered above can also utilise two switches, one of which is `-h` that provides a description of the use and arguments of the command, and another which is `-v` that provides the version of P2RAC.

The P2RAC platform therefore offers support for managing clusters on the Amazon cloud, which includes the creation and termination of single and multiple clusters. Management of data is facilitated by making large and small chunks of data available to the executing task by sharing persistent data volumes and by synchronising data from the Analyst site to the instances of the cluster.

The platform is designed for both batch mode execution and interactive mode execution. Batch-mode execution in P2RAC supports time consuming production tasks. The core commands are listed in a script and the script is executed without the intervention of an Analyst. The interactive mode execution on the other hand allows an Analyst to experiment with his scripts and supports execution of ad hoc tasks. The core commands are executed from the command line by the Analyst.

TABLE I: Preliminary experimental results from P2RAC

Name of cluster	Cluster A	Cluster B	Cluster C	Cluster D
No. of instances	2	4	8	16
No. of virtual cores	8	16	32	64
Avg. time for an optimization cycle (minutes)	120	61	40	29
Total time for the optimization problem (minutes)	3012	1514	1004	728

IV. Workflow & Preliminary Experiment

A workflow of how to use the P2RAC commands is considered in this section. Figure 2 shows the sequence of the commands and the order in which they are executed at the Analyst site for executing a task on the Amazon cloud. An Analyst in possession of R scripts utilizing SNOW library firstly creates a cluster with 4 instances (1 master and 3 worker instances). In this workflow it is assumed that the cluster has the necessary data required by the R scripts (EBS volume). The Analyst secondly sends the R scripts to the cluster that was created. The R script is then executed and the result of execution is generated on the master instance. When the execution of the R scripts are completed then the Analyst gathers the results to his site. The cluster is finally terminated by the Analyst if he does not require to run any more scripts.

In order to validate the feasibility of P2RAC a large-scale sample script, CATopt, that was provided by our industrial partner was employed. CATopt performs a basis risk minimization task using a multi-objective optimization in a high (2000–4000) dimensional space. This optimization, which is useful for the analysis of catastrophe bonds, is performed using the rgenoud package which combines evolutionary search algorithms with derivative-based (Newton or quasi-Newton) methods.

The experiment used the above workflow for executing the CATopt script. Four clusters, namely Cluster A, Cluster B, Cluster C and Cluster D, each with 2, 4, 8 and 16 instances respectively were deployed. The instance are of m2.2xlarge Amazon instance type and therefore 4 virtual cores are available on each instance. The data for the script was already made available on the clusters. The parameters of the optimization algorithm, firstly the population size was set to 200, secondly the maximum number of generations was set to 50 and thirdly the number of optimization cycles was set to 25. The average time taken for each optimization cycle and the total time taken to complete 25 optimization cycles are presented in Table I.

Figure 3 (left) shows total time in hours as the number of virtual cores is increased and Figure 3 (right) shows the corresponding relative speed-up. We observe near perfect parallel efficiency on up to 16 virtual processors, which then falls to 78% efficiency at 32 cores and 50% efficiency at 64 cores. Basically, as we increase to core count we increase the communication overhead and consequentially see a reduction in parallel efficiency. While we might expect to see better

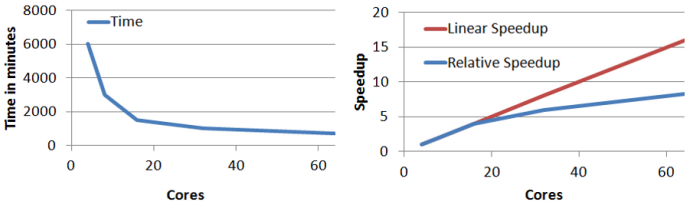

Fig. 3: left: Graph plotted for the number of cores vs total time for the optimization problem; right: Graph plotted for the number of cores vs relative speed-up for the optimization problem

efficiency running on a dedicated high-performance cluster, the achieved performance is quite satisfactory given (a) the low cost of the infrastructure and (b) no special work went into tuning the code and which went straight from an Analysts workstation to running on the cloud.

V. Conclusion & Future Work

A Platform for Parallel R-based Analytics on the Cloud Infrastructure (P2RAC) has been presented in this paper. The platform is developed to support an Analyst who needs to exploit the computing and storage potential of the cloud infrastructure. However, in most cases an Analyst is less likely to know the technicalities of the cloud computing infrastructure. To this end, the P2RAC platform plays the essential role of bridging the gap between the Analyst project and the cloud computing infrastructure. P2RAC offers a set of core and diagnostic tools in the form of commands which enable an Analyst to manage clusters on the cloud, to manage both large and small chunks of data and execute R-based scripts on the cloud. The experimental studies have confirmed the feasibility of employing P2RAC for R-based analytics.

Future work will include the extension of the platform onto private clusters built on cloud using Eucalyptus [26] and OpenNebula [27]. Efforts will be made to offer commands that will enable an Analyst to scale the cluster size by allowing him to add worker instances. Immediate efforts will be made to study how Simple Storage Service (S3) can be incorporated for data management. Spot instances for cost effectiveness offered by Amazon which are an alternative to conventional compute and storage instances will be explored. An evaluation of the performance of P2RAC compared against other high-performance computing infrastructures will be pursued and reported elsewhere.

Acknowledgment

The authors would like to thank Dr. Georg Hoffman and Dr. Oliver Baltzer of Flagstone RE, Halifax, Canada for their support and participation in this research. We would also like to thank Amazon AWS in Education Grant Program for providing access to computational resources for this research.

References

[1] C. Evangelinos, P. F. J. Lermusiax, J. Xu, P. J. Haley, Jr, and C. N. Hill, “Many Task Computing for Real-Time Uncertainty Prediction and Data Assimilation in the Ocean”, IEEE Transactions on Parallel and Distributed Systems, Volume 22, Issue 6, 2011, pp. 1012-1024.

[2] C. Zhang, H. De Sterck, A. Aboulnaga, H. Djambazian and R. Sladek, “Case Study of Scientific Data Processing on a Cloud Using Hadoop”, Proceedings of the 23rd International Conference on High Performance Computing Systems and Applications, 2010, pp. 400-415.

[3] C. Vecchiola, S. Pandey and R. Buyya, “High-Peformance Cloud Computing: A View of Scientific Applications”, Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms and Networks, 2009, pp. 4-16.

[4] R. Anthony, J. Fritz and D. Barnhart, “Cloud Computing Applications for Large-Scale Satellite Ground Systems”, Proceedings of the Military Communications Conference, 2011, pp. 1894-1898.

[5] J. Li, M. Humphrey, Y. -W. Cheah, Y. Ryu, D. Agarwal, K. Jackson and C. van Ingen, “Fault Tolerance and Scaling in e-Science Cloud Applications: Observations from the Continuing Development of MODISAzure”, Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium, Atlanta, USA, 2010, pp. 246-253.

[6] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman and J. Good, “On the Use of Cloud Computing for Scientific Workflows”, Proceedings of the 4th IEEE International Conference on eScience, USA, 2008, pp. 640-645.

[7] Y. Xiaoqiang and D. Yuejin, “Exploration of Cloud Computing Technologies for Geographic Information Services”, Proceedings of the 18th International Conference on Geoinformatics, 2010, pp. 1-5.

[8] S. Kelly, C. Mazyck, K. Pfeiffer and M. -T. Shing, “A Cloud Computing Application for Synchronized Disaster Response Operations” Proceedings of the IEEE World Congress on Services, 2011, pp. 612-616.

[9] W. N. Venables, D. M. Smith and the R Development Core Team, “An Introduction to R”, Notes on R: A Programming Environment for Data Analysis and Graphics, Version 2.14.2, 2012.

[10] A. Quarteroni and F. Saleri, “Scientific Computing with MATLAB and Octave”, Springer, 2nd Edition, 2006.

[11] M. J. Perez, “Multi-Objective Optimization Evolutionary Algorithms in Insurance-Linked Derivatives”, in ‘Handbook of Research on Nature Inspired Computing for Economics and Management’, Edited by J. -P. Rennard, IGI Global, 2007, pp. 885-908.

[12] P. Grossi and H. Kunreuther, “Catastrophe Modeling: A New Approach to Managing Risk”, Springer, 1st Edition, 2005.

[13] K. Chine, “Scientific Computing Environments in the Age of Virtualization, Toward a Universal Platform for the Cloud” , Proceedings of the IEEE International Workshop on Opensource Software for Scientific Computation, 2009, pp. 44-48.

[14] CycleCloud Website: http://cyclecomputing.com/cyclecloud/overview

[15] Cloud Foundry Website: http://www.cloudfoundry.com/

[16] StarCluster Website: http://web.mit.edu/star/cluster/

[17] P. Pacheco, “An Introduction to Parallel Programming”, Morgan Kaufmann, 1st Edition, 2011.

[18] Simple Network of Workstations (SNOW) website: http://www.sfu.ca/ sblay/R/snow.html

[19] N. Leavitt, “Is Cloud Computing Really Ready for Prime Time?”, IEEE Computer, Volume 42, Issue 1, January 2009, pp. 15-20.

[20] Amazon Elastic Compute Cloud (EC2) website: http://aws.amazon.com/ec2/

[21] Amazon Elastic Block Store (EBS) website: http://aws.amazon.com/ebs/

[22] Amazon Machine Images (AMI) website: http://aws.amazon.com/amis

[23] Bioconductor AMI website: http://bioconductor.org/help/bioconductor-cloud-ami/

[24] W. R. Mebane, Jr. and J. S. Sekhon, “Genetic Optimization Using Derivatives: The rgenoud Package for R”, Journal of Statistical Software, Volume 42, Issue 1, May 2011.

[25] Amazon Simple Storage Service (S3) website: http://aws.amazon.com/s3/

[26] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff and D. Zagorodnov, “The Eucalyptus Open-Source Cloud-Computing System”, Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Washington, USA, 2009, pp. 124-131.

[27] C12G Labs, “Private Cloud Computing with OpenNebula 1.4”, 2010.