Accelerating R-based Analytics on the Cloud†

Ishan Patel, Andrew Rau-Chaplin and Blesson Varghese*

Risk Analytics Laboratory, Faculty of Computer Science
Dalhousie University, Halifax, Nova Scotia, Canada

SUMMARY

This paper addresses the problem of harnessing cloud-based infrastructure for the kind of analytical workloads that abound in application domains ranging from computational finance and risk analytics to engineering and manufacturing. Often in this setting, software is not developed by a professional programmer, but on an ad hoc basis by Analysts in high-level programming environments such as R or Matlab. The goal is to allow Analysts to take an analytical job, including both the software and associated data that executes on their personal workstations, and with minimum effort execute the software on a large-scale parallel cloud infrastructure and manage both the resources and the data required by the job. If this can be facilitated gracefully, then the Analyst can not only benefit from experimenting with large-scale analytical problems in less time but also from on-demand resources, low maintenance cost and scalability of computing resources, all of which are offered by the cloud.
In this paper, a Platform for Parallel R-based Analytics on the Cloud (P2RAC) that is placed between an Analyst and a cloud infrastructure is proposed and implemented. P2RAC offers a set of command-line tools for managing the resources, such as instances and clusters, the data and the execution of the software on the Amazon Elastic Computing Cloud infrastructure. Experimental studies are pursued on two parallel problems and the results obtained confirm the feasibility of employing P2RAC for solving large-scale analytical problems on the cloud. Copyright © 2010 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: cloud computing; data analytics; catastrophe bonds; R script

1. 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. Applications in climate simulation and analysis [1], biomedical image processing [2], satellite data processing [3], astronomy [4], and disaster response systems [5] have all be successfully tackled using cloud infrastructure. Such successes in both the commercial and scientific settings have depended on bringing to bear the talents of computer scientists and expert developers 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 [6], Matlab [7] or Octave [7], might be supported in harnessing the cloud for ad hoc analytical workloads.

Prepared using cpeauth.cls [Version: 2010/05/13 v3.00]

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 often involve simulation [8] and optimization [9] 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/change 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 [10][11][12][13] 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 EC2 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 **P**latform for **P**arallel **R**-based **A**nalytics on **C**loud Infrastructure (P2RAC) has been designed to harnessing on-demand compute and storage resources available on the cloud for analytical workloads, and at the same time simplify an Analyst's use of the cloud infrastructure. It provides an interface/API for Analysts to 1) Set-up both individual machines and clusters of machines in the cloud, 2) Associate with machines both persistent and short term storage, 3) Transfer both code and data 4) Execute both batch and interactive job, and 5) Manage execution and resource termination. The goal is to allow Analysts to submit jobs to the cloud with minimum effort, and manage both the resources and the data required to solve a problem and execute it on the cloud. The current implementation of P2RAC provides support for the resources available on the Amazon AWS infrastructure and offers a set of both core and diagnostic tools, but could also be implemented on other cloud platforms.

In our interaction with Analysts working in industry we have observed a number of commonly arising work patterns. In each of these an Analysts starts by prototyping an analytical code on their personal workstations and then reaches a point where they require more computing resources. The need for additional computational resources may take a number of forms, including 1) needing to perform production runs on a single workstation with more memory or compute speed, 2) needing to perform a parameter sweep in which the same code is run hundreds or thousands of times with different input parameters, or 3) needing to speed-up a single long running optimization or simulation task by exploiting co-operative parallelism that may be built into the the optimization or simulation library being used. P2RAC has been designed to address each of these cases. In the later two cases it is important to balance easy-of-use with parallel speed-up. The experimental section of this paper explores this trade-off and shows that reasonable speed-up can often be obtained on cloud infrastructure without undue complexity.

The remainder of this paper is organized as follows. Section 2 shows the sequence of commands for executing an analytical task on cloud resources. Section **??** considers the design of P2RAC and the resources employed by it. Section 3 presents the implementation of P2RAC and the set of commands offered by the platform. Section 5 describes two analytical problems that are executed on the cloud using P2RAC and the results obtained from the experiments. Section 6 concludes the paper.

2. P2RAC: DESIGN OVERVIEW

In this section we discuss the primary goals that guided the design of P2RAC, its software structure, and its usage.

2.1. Design Goals

Our focus in designing P2RAC was to provide Analysts with a platform that would allow them to exploit cloud based resources in a manner that was compatible with their fast moving and data intensive work environment.

2.1.1. Support for how Analysts work Analysts are typically interested in the results of an analysis than in the. Security: Our primary goal was to support Analysts. Minimum difference between workstation and cloud computations.

2.1.2. Support for a variety of parallel abstractions that supports typical analysts work:

2.1.3. Support for a variety of data storage models

2.1.4. Support for result and error logging , Robustness:, Support for results logging:

- Use of standards and libraries:
- Easy of deployment:

2.2. Software Platform and Usage

This section considers how the platform fits in coherently between an Analyst, who is the typical user and the cloud infrastructure (refer Figure 1). The Analyst's project comprising R scripts and large data files is passed on to P2RAC as a task for execution on the cloud. The platform then 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 resources and manages task execution. After the task is executed, the platform gathers results which may be spread across the resources in the cloud.

The platform provides support both for an individual computing resource as well as for clusters (a collection of computing resources that work collaboratively). The platform comprises three components, (a) the core tools, (b) the diagnostic tools and (c) the configuration files. The core tools provide functionalities for resource 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.

Two typical usage scenarios: batch and interactive....

Interactive usage:

Step 1:
Step 2:
Step 3:
Step 4:
Step 5:

3. SYSTEM DESIGN AND IMPLEMENTATION

As considered above P2RAC comprises three components, (i) the core tools, (ii) the diagnostic tools and (iii) the configuration files supporting both instances and clusters. This section considers the implementation of the tools that support the cluster and instance. The core and diagnostic tools are implemented as commands which are executed from the command-line. All commands in P2RAC
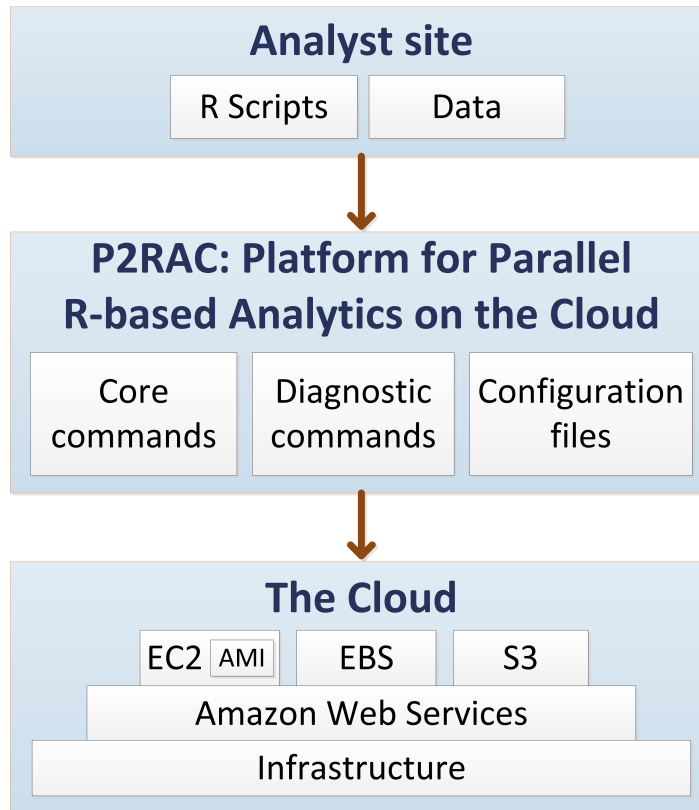
Figure 1. Conceptual design of P2RAC

utilise two switches, one of which is -h that provides a description of the use and arguments of the command, and the other which is -v that provides the version of P2RAC.

3.1. AWS Support

Large scale data analytics is investigated on the cloud infrastructure which is a pool of computational and storage resources geographically dispersed. The resources provided by a cloud are accessible through Infrastructure as a Service (IaaS) [26], and Amazon is a leading provider of IaaS. 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 Amazon instances are initialized using Amazon Machine Images (AMI) [29]. The Bioconductor Cloud AMI [30] was employed during the preliminary test phase of the research. The SNOW package and the R version of GENetic Optimization Using Derivatives (rgenoud) package [31] are additionally installed on the instance when the Bioconductor Cloud AMI (this AMI does not provide SNOW and rgenoud packages) is used. Subsequently, it was necessary to utilise a custom AMI.

Two Ubuntu AMIs are used in this research. The first AMI, supports Cluster Compute instances offered by Amazon. The Cluster Compute instances are built on Hardware Virtual Machine (HVM) type virtualization. The second Ubuntu AMI offers support for non-HVM type virtualization. Any additional libraries which an Analyst needs to include in his project can be installed on the instance by specifying library packages in a configuration file.

The storage resources on Amazon are referred to as the Elastic Block Storage (EBS) [28]. Similar to EC2, EBS is 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 is exploited when large volumes of data need to be used in an Analyst's 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 making data locally available to the instance.

The EC2 and EBS services are employed by P2RAC. The P2RAC platform mediates between an Analyst's 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 instances on the cloud for P2RAC.

3.2. Core Tools

The core tools of P2RAC provide functionalities for resource management, data management on the resource, and execution management of a task on the resource. An interface is provided to an Analyst to access 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.

3.2.1. Instance Support is offered using tools for instance management, data management on the instance and execution management of a task on the instance.

Instance Management tools configure an instance on the cloud, offers the instance for the execution of a task and finally terminates the instance after the task it was executing is completed. P2RAC offers two tools for managing instances, namely `ec2createinstance` and `ec2terminateinstance`.

The `ec2createinstance` command is responsible for configuring an instance on the cloud and making it available to an analyst.

The syntax of the `ec2createinstance` command is

```
ec2createinstance [-h] [-iname INSTANCE_NAME] [-ebsvol
EBS_VOLUME | -snap EBS_SNAP] [-type INSTANCE_TYPE] [-desc
INSTANCE_DESCRIPTION]
```

The optional arguments of the `ec2createinstance` command are `iname`, `ebsvol`, `snap`, `type` and `desc`. `iname` specifies the name of the instance that is created. `ebsvol` specifies the EBS volume ID which is provided by Amazon when an EBS volume is created. `snap` specifies the EBS snapshot ID from which an EBS volume can be created. `snap` and `ebsvol` cannot be specified at the same time. `ebsvol` can be specified when a EBS volume is available, while 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. `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 for a number of experiments in the work reported in this paper. The `desc` argument can be used to provide a description of the instance.

For example, if a command such as

```
ec2createinstance -iname 'hpc_instance' -ebsvol 'vol-xxxxxxxx'
-type 'm2.4xlarge' -desc 'For Trial Simulation Run'
```

is executed, then a sequence of activities follow. One EC2 instance of `m2.4xlarge` type is initialized using the AMI specified in the configuration file, and is tagged. The EBS volume, `vol-xxxxxxxx` (volume ID is masked in this paper) is attached on to the instance. A configuration file at the Analyst site is updated with instance information such as the public DNS names of the instance, EBS volume ID and description of the instance. 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 `ec2createinstance` command facilitates the creation of multiple instances. Since an EBS volume can only be attached to one instance, and therefore, the need for multiple EBS volumes arises when multiple instances 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 [32]. Multiple instances cannot have the same name when the `ec2createinstance` command is executed more than once.

When a task has completed execution on the instance it is essential to safely release the instance. The `ec2terminateinstance` command is provided. The syntax of the `ec2terminateinstance` is

```
ec2terminateinstance [-h] [-v] [-iname INSTANCE_NAME]
[-deletevol]
```

The optional arguments of the `ec2terminateinstance` command are `iname` and `deletevol`. `iname` specifies the name of the instance that needs to be terminated. The `deletevol` switch deletes an EBS volume attached to the instance being terminated. To terminate all the instances on the cloud `ec2terminateall` command is provided which will be considered in cluster management.

For example, if a command such as

```
ec2terminateinstance iname 'hpc_instance'
```

is executed, then a sequence of activities follows. Firstly, the EBS volume attached to the instance is no more available. Further to this, the instance is terminated. The section containing the instance information of `hpc_instance` in the configuration file is removed. Should the `deletevol` switch be included in the command then the EBS volume, `vol-xxxxxxxx` is deleted.

Data Management on the instance is required to transfer the script and the data from the Analyst's site onto the instance and then receive results. The Secure Copy Protocol (SCP) or the rsync protocol can be employed for data transfer. However, rsync, is chosen as it transfers data quicker than SCP. Moreover, rsync in subsequent data transfers only synchronises the data changed at the source. This feature of rsync makes it suitable for P2RAC since data at the Analyst's site changes frequently. The `ec2senddatatoinstance` command is provided for data transfer and the results are retrieved using the `ec2getresultsfrominstance` command.

The syntax of the `ec2senddatatoinstance` command is

```
ec2senddatatoinstance[-h] [-v] [-iname INSTANCE_NAME]
[-projectdir PROJECT_DIRECTORY]
```

The optional arguments of the `ec2senddatatoinstance` command are `iname` and `projectdir`. `iname` specifies the name of the instance to which the project directory will be synchronised. The source project directory is specified as the argument `projectdir`. If the instance name is not provided by the Analyst then the instance 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.

The project directory comprises 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. 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 instance.

The syntax of the `ec2getresultsfrominstance` is

```
ec2getresultsfrominstance [-h] [-v] [-iname INSTANCE_NAME]
[-projectdir PROJECT_DIRECTORY] [-runname RUN_NAME]
```

The optional arguments are `iname` and `projectdir`. The `iname` argument specifies the name of the instance 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 instance. If no project directory is specified then the path of the current working directory at the Analyst site is used.

The mandatory argument for the `ec2getresultsfrominstance` 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.

Execution Management command, namely the `ec2runoninstance` runs the R script on the instance. This command locks the instance onto the R script and does not permit any additional use of the instance until the script has completed execution or the instance is manually unlocked using `ec2resourcelock` considered later. The syntax of `ec2runoninstance` is

```
ec2runoninstance [-h] [-v] [-iname INSTANCE_NAME] [-projectdir
PROJECT_DIRECTORY] [-rscript R_SCRIPT] [-runname RUN_NAME]
```

The optional arguments of `ec2runoninstance` are `iname`, `projectdir` and `rscript`. The `iname` argument specifies the name of the instance on which 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 instance. `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 `ec2runoninstance` is `runname` which indicates the name of a run.

3.2.2. Cluster Support is offered using tools for cluster management, data management on the cluster and execution management of a task on the cluster.

Cluster Management in the core tools are a set of functionalities that range from gathering a pool of instances on the cloud, followed by configuring the instances 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 instances as a cluster on the cloud.

The syntax of the `ec2createcluster` command is

```
ec2createcluster [-h] [-v] [-cname CLUSTER_NAME] [-csize
CLUSTER_SIZE] [-ebsvol EBS_VOLUME | -snap EBS_SNAP] [-type
INSTANCE_TYPE] [-desc CLUSTER_DESCRIPTION]
```

The optional arguments of the `ec2createcluster` command are `cname`, `csize`, `ebsvol`, `snap`, `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. `snap` specifies the EBS snapshot ID from which an EBS volume can be created. `snap` and `ebsvol` cannot be specified at the same time. `ebsvol` can be specified when a EBS volume is available, while 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. `type` defines the Amazon EC2 instance type which is specified based on the computational requirements of the task. 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 AMI specified 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, description of the cluster and whether the cluster is in use for executing a script. 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 [32]. 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 [-h] [-v] [-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, then whether a cluster is in use is firstly checked. If the cluster is in use, then the cluster cannot be terminated. If the cluster is not in use, then a sequence of activities follows. 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.

When all resources, both of the instance and the cluster need to be terminated the `ec2terminateall` command is provided. The syntax of `ec2terminateall` is

`ec2terminateall [-h] [-v] [-instances] [-clusters] [-ebsvolumes] [-snapshots]`

The optional switches are `instances`, `clusters`, `ebsvolumes` and `snapshots` which terminates all the instances, clusters, EBS volumes and snapshots respectively.

Data management on the cluster 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 `ec2senddatatoclusternodes` and `ec2senddatatomaster` based on the rsync protocol are provided. In order to receive the results to the Analyst site the `ec2getresults` command is developed.

The `ec2senddatatoclusternodes` 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 `ec2senddatatoclusternodes` 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 `ec2senddatatoclusternodes` command is

```
ec2senddatatoclusternodes [-h] [-v] [-cname CLUSTER_NAME]
[-projectdir PROJECT_DIRECTORY]
```

The optional arguments of the `ec2senddatatoclusternodes` 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, `ec2senddatatomaster` command is provided.

The syntax of the `ec2senddatatomaster` is

```
ec2senddatatomaster [-h] [-v] [cname CLUSTER_NAME] [-projectdir
PROJECT_DIRECTORY]
```

The optional arguments of `ec2senddatatomaster` are similar to that of `ec2senddatatoclusternodes`.

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 [-h] [-v] [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.

Execution Management assigns the Analyst task onto a cluster and further runs task on the cluster. This command locks the cluster for execution of the R script specified by the execution command and does not permit any additional use of the cluster until the script has completed execution or the cluster is manually unlocked using `ec2resourcelock` considered in the next section. For this, the `ec2runoncluster` command is provided. The syntax of `ec2runscript` is

```
ec2runoncluster [-h] [-v] [-cname CLUSTER_NAME] [-projectdir
PROJECT_DIRECTORY] [-rscript R_SCRIPT] [-runname RUN_NAME] [-bynode
| -byslot]
```

The optional arguments of `ec2runoncluster` 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.

Two optional switches to manage the scheduling of slave processes onto the cores of the cluster nodes are available. The `bynode` switch assigns the processes in a round-robin fashion while the `byslot` switch assigns all processes on a node until all of its cores are exhausted. In MPI, the default scheduling is `byslot`, whereas in P2RAC, `bynode` is chosen as the default scheduling mechanism if the switch is not specified. `bynode` switch is required to meet the memory constraints of large processes.

The mandatory argument for `ec2runoncluster` is `runname` which indicates the name of a run.

3.3. Diagnostic Tools

support instances and clusters and are available as follows:

(a) `ec2listinstances`, `ec2listclusters` and `ec2listallresources` - for listing the instances and clusters created by the Analyst on the Amazon cloud
(b) `ec2logintoinstance` and `ec2logintomaster` - for accessing an instance or the master instance of a cluster
(c) `ec2resoucelock` - to lock an instance or a cluster for a specific task or to unlock them from use

The syntax of `ec2listinstance` is

```
ec2listinstance [-h] [-v] [-names]
```

The optional switch `names` provides the names of the instance. If the switch is not provided then the list of the instances, their public DNS names, volume ID of the EBS volume shared with the instances and the description of the cluster is provided.

The syntax of `ec2listclusters` is

```
ec2listclusters [-h] [-v] [-names]
```

The optional switch `names` 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 `ec2listallresources` is

```
ec2listallresources [-h] [-v] [-instances] [-ebsvols] [-snapshots]
[-amis]
```

The switches `instances`, `ebsvols`, `snapshots` and `amis` provides the names of the instances, EBS volumes, EBS snapshots and AMIs on the cloud.

The syntax of `ec2logintoinstance` is

```
ec2logintoinstance [-h] [-v] [-iname INSTANCE_NAME]
```

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

The syntax of `ec2logintocluster` is

```
ec2logintocluster [-h] [-v] [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 also 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.

The syntax of `ec2resourcelock` is

```
ec2resourcelock [-h] [-v] [-iname INSTANCE_NAME | cname
CLUSTER_NAME] [-free | -inuse]
```

The optional arguments `iname` specifies the name of the instance and `cname` the name of the cluster that needs to be locked or unlocked. The resources are locked using the `-inuse` switch and unlocked using the `-free` switch.

3.4. Configuration Files

There are four 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 provide support for instances, and includes the name of the instance created, its public DNS name, Volume ID, the description of the instance and whether the instance is in use. Thirdly, a file that provides support for clusters, and includes the names of the clusters created, their size, the public DNS names of all their instances, Volume ID of the EBS volume shared by the master with the workers, the description of the clusters and whether the cluster is in use. Fourthly, a file that contains a list of R libraries which are required by an Analyst's 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.

The P2RAC platform offers support both for instances and clusters. In the case of instance support, P2RAC enables the management of instances, which includes the creation and termination of single and multiple instances. In the case of cluster support, P2RAC facilitates creation and termination of single and multiple clusters. In either case, 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 or 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.

4. EXAMPLE WORKFLOWS

A workflow of how to use the P2RAC commands for executing a task on instances or on a cluster is considered in this section. Firstly, for running an R-script on an instance, the sequence of commands
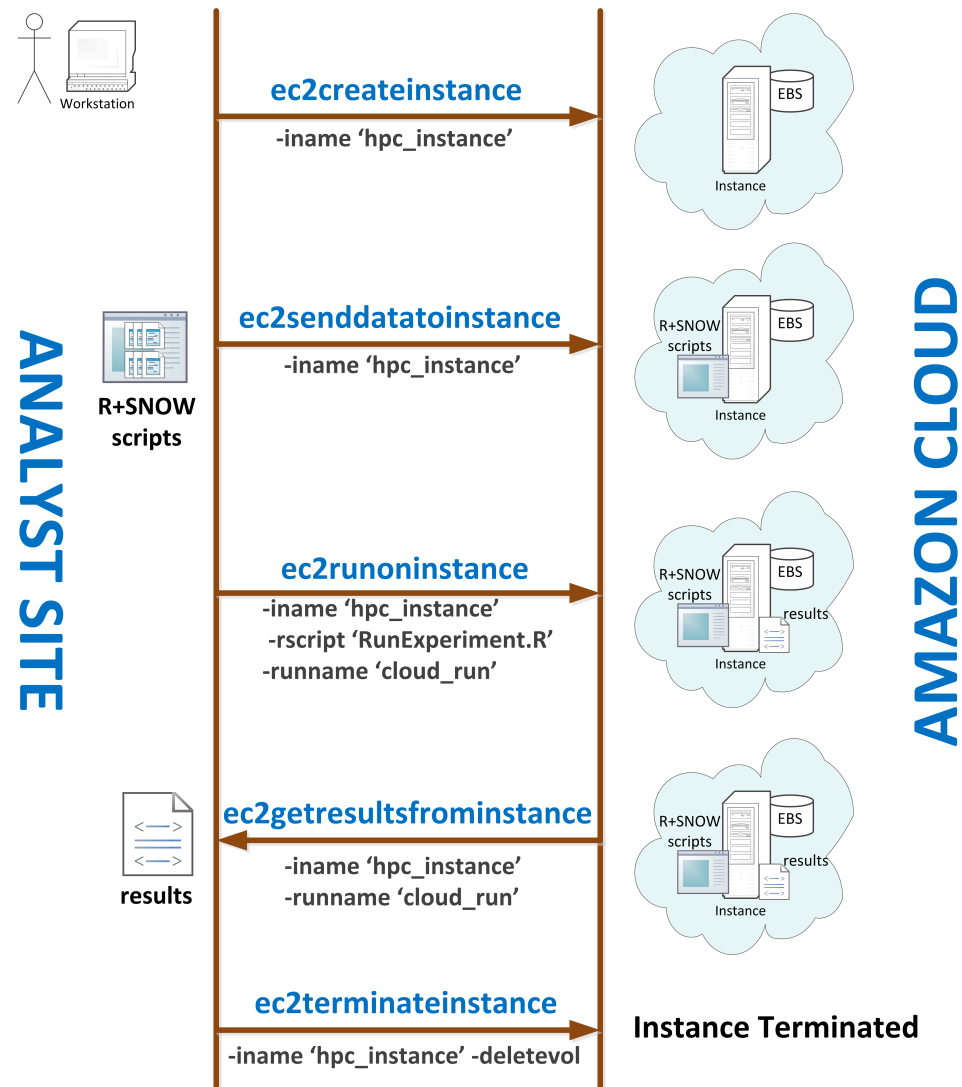
Figure 2. Workflow of commands using P2RAC

are shown in Figure 2. An Analyst in possession of R scripts utilizing SNOW library firstly creates an instance on the Amazon Cloud. The scripts and the data required by the script are then provided to the instance. The script is executed on the instance and produces the results on the instance. The results can then be fetched by the Analyst. Finally, the instance is terminated.

Figure 3 shows the sequence of the commands and the order in which they are executed at the Analyst site for executing a task on the cluster. A cluster with 4 instances (1 master and 3 worker instances) is created. The Analyst secondly sends the R scripts and the data required by the scripts to the cluster. The 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.

5. EXPERIMENTAL STUDIES

This section presents the platform, the two problems employed on P2RAC and the results obtained from experiments.
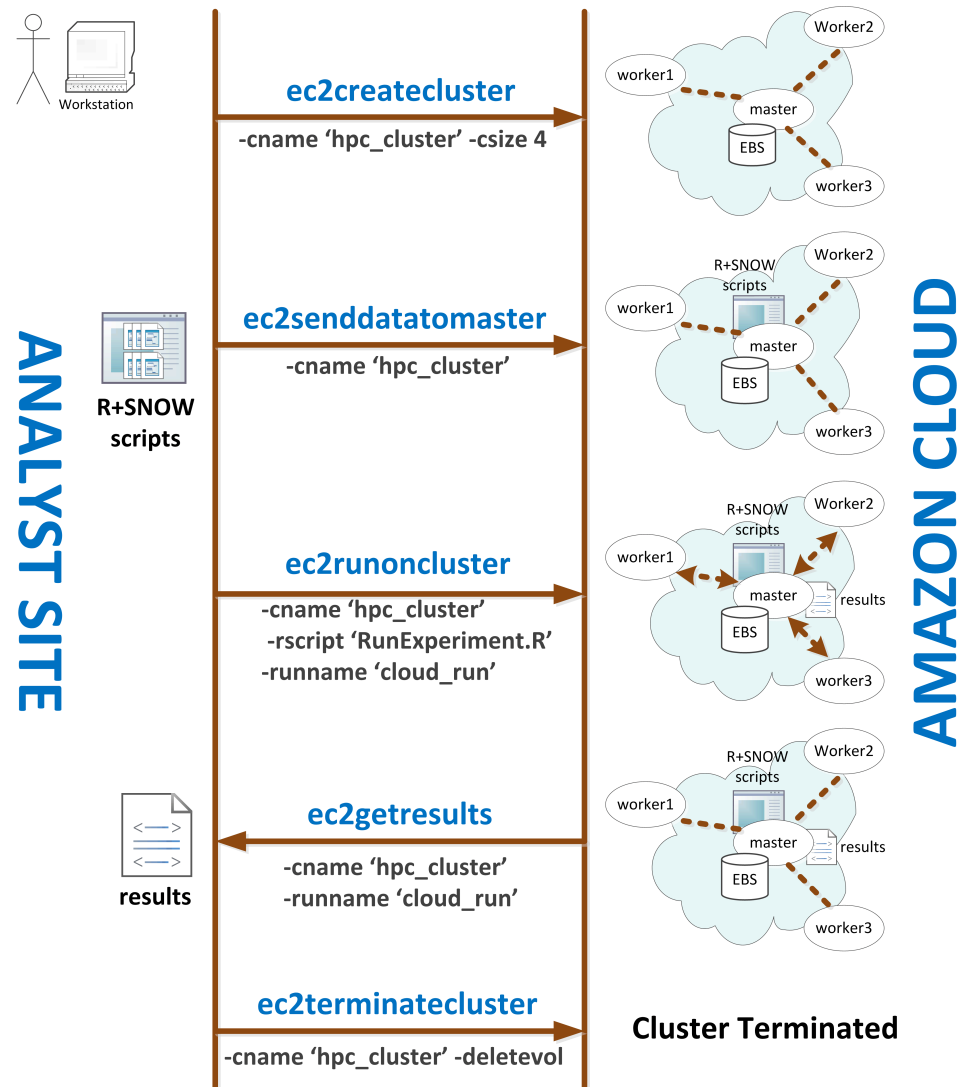
Figure 3. Workflow of commands using P2RAC

5.1. Experimental Platform

Two desktop computers, two Amazon cloud instances, and four clusters comprising Amazon cloud instances are the computational resources utilised for the evaluation of P2RAC. Table I shows the specifications of the resources.

5.2. Experimental Problems

To evaluate the feasibility of P2RAC, two kinds of experimental problems, which are analytical in nature were employed. The first problem is a computationally intensive task employing co-operative parallelism. An R-based example of such a task was used, which is referred to as Catastrophe Bond Optimization (CATopt). 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 input data to the CATopt problem is approximately 300 MB.

Table I. Resources Utilised for Experimental Studies

| Resource	Provided by	Processor / Amazon Type	No. of cores	Memory	Storage	System type
Desktop A	Dalhousie University	Intel (R) Core (TM) i7-2600 @ 3.4 Ghz	8	16GB	1.8TB	64 bit
Desktop B	Flagstone Re		8			64 bit
Instance A	Amazon	m2.2xlarge	4	34.2 GB	850 GB	64 bit
Instance B	Amazon	m2.4xlarge	8	68.4 GB	1690 GB	64 bit
Cluster A	Amazon	m2.2xlarge X 2	8	68.4 GB	1.7 TB	64 bit
Cluster B	Amazon	m2.2xlarge X 4	16	136.8 GB	3.4 TB	64 bit
Cluster C	Amazon	m2.2xlarge X 8	32	273.6 GB	6.8 TB	64 bit
Cluster D	Amazon	m2.2xlarge X 16	64	547.2 GB	13.6 TB	64 bit
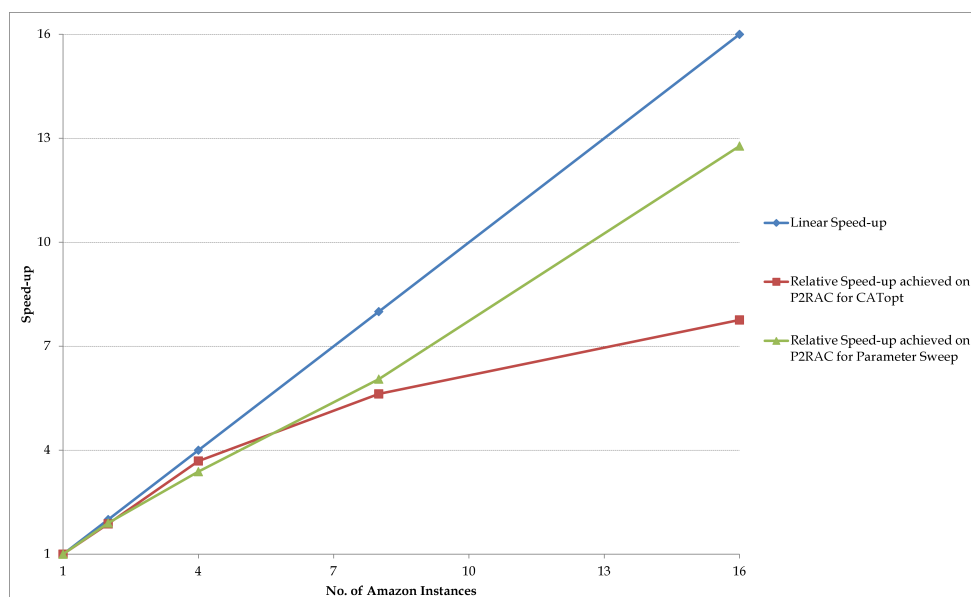
Figure 4. Speed-up achieved for the CATopt and Parameter Sweep Problems using P2RAC

The second problem is a parameter sweep task that runs multiple independent jobs. An R-based example of such a task was used. The input data to the parameter sweep task is approximately only 3 MB.

5.3. Experimental Results

Figure 4 is a graph showing the relative speed-up achieved for both the experimental problems with increasing number of Amazon instances. Figure 5 is a bar graph that shows the timing for the best results on the two desktops, the two Amazon instances and the four Amazon clusters. The best performance is achieved on Cluster D.

For the CATopt problem in these experiment, the population size is set to 200 and the maximum generations is set to 50. In both the problems there is a near 100% efficiency for upto 4 Amazon instances, after which there is a drop in the parallel efficiency. The reduction in parallel efficiency is due to increase in communication overheads between virtualised cloud instances. The acceleration achieved for both the problems is satisfactory considering the low cost of the infrastructure
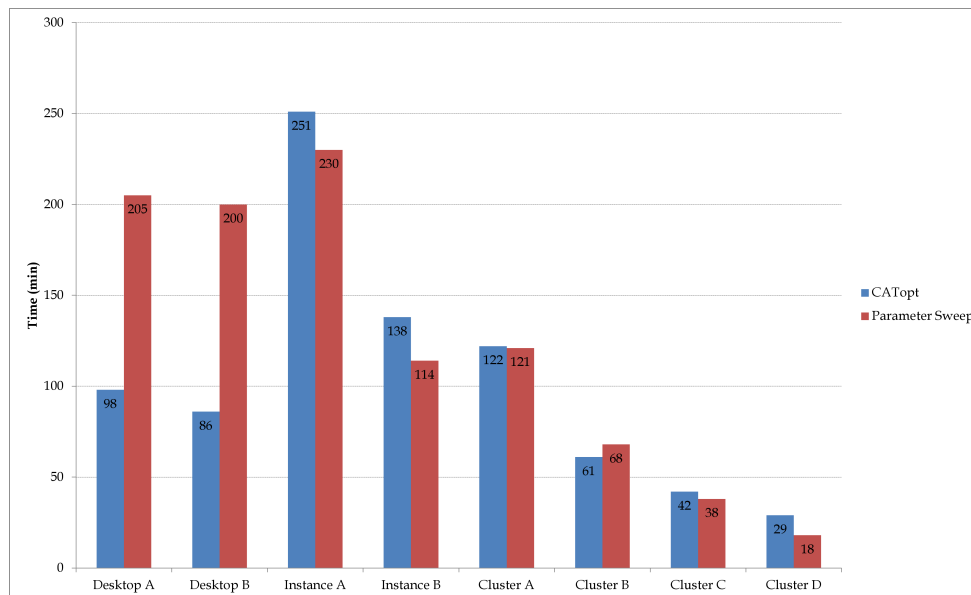
Figure 5. Best-case timing results of the CATopt and Parameter Sweep Problems using P2RAC

employed and that the problems can be directly deployed from an Analyst's site without any additional tuning.

6. CONCLUSION AND FUTURE WORK

Analytical jobs can benefit from harnessing the benefits of cloud computing such as on-demand availability of resources, scalability of the resources and low costs for maintenance. A platform that (i) provides an Analyst with an interface to seamlessly submit an analytical job and collect its results, and (ii) provides the flexibility to manage the resources and jobs is ideal for an Analyst who needs to exploit the benefits of the cloud.

The Platform for Parallel R-based Analytics on the Cloud (P2RAC) proposed and implemented in this paper provides an interface for an Analyst who needs to submit a R-based analytical job on the cloud. P2RAC is therefore placed in between an Analyst and the cloud infrastructure. P2RAC provides support for instances and clusters on the Amazon cloud and offers a set of core and diagnostic tools which can be used from the command-line.

The core tools support resource management, data management on the resource and execution management. Resource management functionalities range from gathering resources required for an analytical job and releasing them after their use. Data management ranges from providing the resources with data required for executing an R-script and gathering results from the resources onto the Analyst's site once the R-script has completed execution. Execution management provides functionalities for executing an R-script on the resources acquired.

The diagnostic tools support the testing of resources and provide information of the resources employed by an Analyst. This is facilitated by providing access to log in to the acquired resources.

The feasibility of P2RAC is validated by considering two analytical problems. The algorithm for solving the first problem incorporates co-operative parallelism for optimization, while in the second problem, multiple independent jobs are employed for parameter sweep. Both the problems are provided from an Analyst work site to the Amazon cloud using P2RAC. The results obtained from P2RAC is an evidence that R-based analytical problems can benefit from cloud computing and P2RAC facilitates the execution of the analyst job on the cloud.

The installation package for P2RAC can be obtained from http://www.p2racsoftware.com. The installation is built for the Linux OS, supports Virtual Python Environment [33] and can be installed

on a desktop using Easy Install [34]. After installation, P2RAC can be configured automatically using `ec2configurep2rac`.

In the future, it is anticipated that another version of P2RAC will be released that incorporates the support for spot instances. Fault tolerance through distributed checkpointing for spot instance used in P2RAC will be explored. Further, the dynamic scaling of clusters, i.e., increase and decrease the size of a cluster when required by a job that is executing on the cluster, will be investigated for implementation.

ACKNOWLEDGEMENT

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. Zhang C., De Sterck H., Aboulnaga A., Djambazian H. and Sladek R. 2010. *Case Study of Scientific Data Processing on a Cloud Using Hadoop*. Proceedings of the 23rd International Conference on High Performance Computing Systems and Applications, pp. 400-415.
3. Li J., Humphrey M., Cheah Y.-W., Ryu Y., Agarwal D., Jackson K. and Van Ingen C. 2010. *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, pp. 246-253.
4. Hoffa C., Mehta G., Freeman T., Deelman E., Keahey K., Berriman B. and Good J. 2008. *On the Use of Cloud Computing for Scientific Workflows*. Proceedings of the 4th IEEE International Conference on eScience, USA, pp. 640-645.
5. Kelly S., Mazyck C., Pfeiffer K. and Shing M.-T. 2011. *A Cloud Computing Application for Synchronized Disaster Response Operations*. Proceedings of the IEEE World Congress on Services, pp. 612-616.
6. Venables W.N., Smith D.M. and the R Development Core Team. 2012. *An Introduction to R*. Notes on R: A Programming Environment for Data Analysis and Graphics, Version 2.14.2.
7. Quarteroni A. and Saleri F. 2006. *Scientific Computing with MATLAB and Octave*. Springer, 2nd Edition.
8. Grossi P. and Kunreuther H. 2005. *Catastrophe Modeling: A New Approach to Managing Risk*. Springer, 1st Edition.
9. Perez M.J. 2007. *Multi-Objective Optimization Evolutionary Algorithms in Insurance-Linked Derivatives*. Handbook of Research on Nature Inspired Computing for Economics and Management, Edited by Rennard J.-P., IGI Global, pp. 885-908.
10. Chine K. 2009. *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, pp. 44-48.
11. CycleCloud website: http://cyclecomputing.com/cyclecloud/overview [Last checked: 10th August 2012]
12. Cloud Foundry Website: http://www.cloudfoundry.com/ [Last checked: 10th August 2012]
13. StarCluster Website: http://web.mit.edu/star/cluster/ [Last checked: 10th August 2012]
14. Iosup A., Ostermann S., Yigitbasi M.N., Prodan R., Fahringer T. and Epema D.H.J. 2011. *Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing*. IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 6.
15. Habich D., Lehner W., Richly S. and Assmann U. 2010. *Using Cloud Technologies to Optimize Data-Intensive Service Applications*. Proceedings of the 3rd IEEE International Conference on Cloud Computing, pp. 19-26.
16. Barga R.S., Ekanayake J. and Lu W. 2012. *Project Daytona: Data Analytics as a Cloud Service*. Proceedings of the 28th IEEE International Conference on Data Engineering, pp. 1317-1320.
17. Nae V., Iosup A. and Prodan, R. 2011. *Dynamic Resource Provisioning in Massively Multiplayer Online Games*. IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 3.
18. Lu S., Li R.M., Tjhi W.C., Lee K.K., Wang L., Li X. and Ma D. 2011. *A Framework for Cloud-Based Large-Scale Data Analytics and Visualization: Case Study on Multiscale Climate Data*. Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science, pp. 618-622.
19. Tancer J. and Varde A.S. 2011. *The Deployment of MML for Data Analytics over the Cloud*. Proceedings of the 11th IEEE international Conference on Data Mining Workshops, pp. 188-195.
20. Kim H., Chaudhari S., Parashar M. and Marty C. 2009. *Online Risk Analytics on the Cloud*. Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and Grid, pp. 484-489.
21. Xue W., Shi J. and Yang B. 2010. *X-RIME: Cloud-Based Large Scale Social Network Analysis*. Proceedings of the IEEE International Conference on Services Computing, pp. 506-513.
22. Marozzo F., Talia D. and Trunfio P. 2011. *A Cloud Framework for Parameter Sweeping Data Mining Applications*. Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science, pp. 367-374.
23. Watson P., Hiden H. and Woodman S. 2010. *e-Science Central for CARMEN: Science as a Service*. Concurrency and Computation: Practice and Experience, Vol. 22, pp. 2369-2380.
24. Pacheco P. 2011. *An Introduction to Parallel Programming* (1st Edition). Morgan Kaufmann.
25. Simple Network of Workstations (SNOW) website: http://www.sfu.ca/ sblay/R/snow.html [Last checked: 8th August 2012]
26. Leavitt N. 2009. *Is Cloud Computing Really Ready for Prime Time?* IEEE Computer, Vol. 42, Issue 1, pp. 15-20.
27. Amazon Elastic Compute Cloud (EC2) website: http://aws.amazon.com/ec2/ [Last checked: 10th August 2012]
28. Amazon Elastic Block Store (EBS) website: http://aws.amazon.com/ebs/ [Last checked: 10th August 2012]
29. Amazon Machine Images (AMI) website: http://aws.amazon.com/amis [Last checked: 10th August 2012]
30. Bioconductor AMI website: http://bioconductor.org/help/bioconductor-cloud-ami/ [Last checked: 10th August 2012]
31. Mebane, Jr. W. R. and Sekhon J. S. 2011. *Genetic Optimization Using Derivatives: The rgenoud Package for R*. Journal of Statistical Software, Vol. 42, Issue 1.
32. Amazon Simple Storage Service (S3) website: http://aws.amazon.com/s3/ [Last checked: 10th August 2012]
33. Virtual Python Environment website: http://pypi.python.org/pypi/virtualenv/ [Last checked: 10th August 2012]
34. Python Easy Install website: http://packages.python.org/distribute/easy_install.html [Last checked: 10th August 2012]