Support for data-intensive computing with CloudMan

Y. Kowsar[1] and E. Afgan [1,2]
[1] Victorian Life Sciences Computation Initiative (VLSCI), University of Melbourne, Melbourne, Australia
[2] Centre for Informatics and Computing (CIR), Ruđer Bošković Institute (RBI), Zagreb, Croatia
ykowsar@student.unimelb.edu.au, enis.afgan@[unimelb.edu.au, irb.hr]

Abstract - Infrastructure-as-a-Service (IaaS) compute infrastructure model has showcased its ability to transform how access to compute resources is realized; it delivered on the notion of Infrastructure-as-Code and enabled a new wave of compute adaptability. However, many workloads still execute only in a more structured and traditional cluster computing environment where jobs are handed off to a job manager and possibly executed in parallel. We have been developing CloudMan (usecloudman.org) as a versatile solution for enabling and managing compute clusters in cloud environments via a simple web interface or an API. In this paper, we describe a recent extension of CloudMan to add support for data intensive workloads by incorporating Hadoop and HTCondor job managers and thus complement the previously available Sun Grid Engine (SGE).

I. Introduction

As the information age continues, the rate at which data is produced is continuing its exponential growth; it is fueled by everything from news services and social media activities to sensor networks and research-driven devices. Recent advances in biology and the advent of Next Generation Sequencing (NGS) systems is a prime example of such developments. This data growth has led to what is often referred to as the data deluge and has posed a shift in the research problems at hand [1]. Several years ago, the rate at which, specifically biological, data was produced was a limiting research factor. Namely, biologists would generate genomic sequences one nucleotide at a time and this allowed the data to be easily stored and analyzed even on a researcher's personal computer.

With the availability of NGS systems those days have been surpassed. Today, it is possible to sequence an entire human genome, consisting of ~3 billion nucleotides, in just a few days, at better quality, and at a fraction of cost compared to just a few years ago. This ability to generate so much data has, however, led to major challenges when dealing with storage and the ability to analyze it.

Although often primarily described as a challenge and an obstacle, the reality is that the availability of the increasing data volume presents enormous opportunities. The real power of the data will not come just from the sheer volume, but from the ability to analyze it. It is thus vital to provide flexible yet accessible solutions that enable researchers and companies alike to move beyond the data collection and step into the world of data analysis.

As a step in this direction, we have been developing a cloud resource manager called CloudMan [2] that facilitates creation of a compute platform [3] capable of handling a range of workloads, including biological analyses [4]. Cloud computing in general allows compute and storage resources to be requested, provisioned, and utilized to handle the necessary scaling of a computational problem at hand. However, those resources are often provisioned as bare virtual machines and disks without application context or coordination. CloudMan helps in this regard by orchestrating all the steps required to provision a functional compute and application environment based on the flexible cloud resources and provides a user with an accessible, functional, and flexible data analysis platform.

In this paper, we describe recent advances in CloudMan where support for data intensive workloads has been added. Until now, CloudMan provisioned a functional and scalable Sun Grid Engine (SGE) compute cluster in the cloud as well as a range of bioinformatics tools as part of CloudBioLinux [5] and Galaxy [6], [7], [8]. We have now added support for Hadoop [9] and HTCondor [10] based workloads, enabling CloudMan to be used as a big-data-platform-on-demand. These advances empower the next wave of data analysis workloads to be seamlessly executed in the cloud environment and easily integrated with the already existing and more traditional pipelines. Support for these types of workloads also facilitates easier development of Hadoop-based tools (due to the accessibility of the cloud and the functional execution/development environment) and provision groundwork for federation of clusters across multiple clouds and/or data centers. It is worth explicitly stating that CloudMan is not limited to biological workloads and can just as readily be utilized for any other workload where a compute cluster is necessary. In the rest of the paper, implementation details and sample use cases for Hadoop and HTCondor via CloudMan are presented.

II. Background

A. Big Data: Hadoop

Big data is a concept introduced as a result of recent explosion of available data in various fields. This also applies to sciences that were traditionally not considered data-intensive and is having a ripple effect on how research is being transformed. One possible approach for tackling big data problems comes in the form of distributed and cloud computing where use of commodity

This work was, in part, supported by the Genomics Virtual Laboratory (GVL) grant from the National eResearch Collaboration Tools and Resources (NeCTAR).

computers enables feasible and accessible solution for a variety of problem domains. These approaches are providing opportunities for researchers and companies alike to store, retrieve and interpret their data in a reasonable amount of time and budget. Hadoop is an example of such a solution; it is a framework for analyzing big data across clusters of computers using a simple programming model. In this model a problem is divided into several independent smaller problems (called *map*), where each sub-problem can be solved over a single node of a cluster. The process of solving the problem continues by collecting the results from each sub-problem to be merged together (called *reduce*) to shape a single output. Programs written in this style can be automatically parallelized and executed on a large cluster of commodity machines [9]. Combining this computation model with the readily available computing infrastructure and making it all accessible will go a long way toward tackling the big data problem.

B. *Federated Computing: HTCondor*

Complementing the notion of distributed computing is the notion of federated computing. In this model, the aim is to assemble distributed computing resources that are possibly geographically and administratively disparate into a cooperating unit. Joining of such resources makes it possible to achieve higher workload throughput and resource utilization; the aim is to utilize all the known and available resources available over a period of time without assuming high availability of those resources [11]. Joining a shared pool of resources brings other complexities such as enforcing each organization's policy and maintaining a service layer agreements. In these scenarios, flexibility is a key for managing complex system. Although traditionally utilized across a dedicated set of resources, it is possible to envision use of federated computing across multiple clouds and/or clouds and dedicated resources. Enabling application execution environments and consequently applications to run in such environments would help minimize vendor lock-in, increase code portability, and create an equilibrium market.

Several projects exist in the field of data-intensive and/or federated computing atop cloud resources; the ones most relevant to the work being presented in this paper are presented in Table 1.

CloudMan brings forward parts of each of the projects listed in Table 1 and yet delivers a solution that caters to users not supported by any of the existing projects. At its core, CloudMan is an accessible cloud manager that establishes functional application execution environments. It operates on a range of cloud providers (namely, AWS, OpenStack, OpenNebula, and Eucalyptus), thus making it the most versatile cloud manager available. Built with CloudBioLinux and Galaxy, it provides ready access to over 100 bioinformatics tools and hundreds of gigabytes of biological reference datasets that can be utilized via Galaxy's web interface, Linux remote desktop, or the command line interface. Nonetheless, CloudMan is not limited to biological workloads and can also be utilized for any workload where a compute cluster is necessary. CloudMan can be installed (via automation; see CloudBioLinux) on any Ubuntu machine image, thus instantly turning that machine image into a functional cloud cluster. Alternatively, prebuilt CloudMan images can be started without any installation via a single web-form (http://biocloudcentral.org). Once launched, CloudMan provides a web interface for managing cloud and cluster resources. Integrating Hadoop and HTCondor into CloudMan extends the already extensive set of provided features and provides support for the new wave of applications and use cases.

TABLE I. LIST AND OVERVIEW OF PROJECTS RELATED TO THE WORK PRESENTED HERE

StarCluster [12]	StarCluster is an open source project created by STAR group at MIT that allows anyone to easily create and manage their own cluster computing environments hosted on Amazon's Elastic Compute Cloud (EC2) without needing to be a cloud expert. The project also provides support for Hadoop based workloads.
Amazon Elastic MapReduce [13]	Amazon's Elastic MapReduce (EMR) is a general-purpose web service that utilizes a hosted Hadoop framework running on Amazon EC2 and Amazon Simple Storage Service (S3) allowing anyone to process data-intensive workloads on Amazon Web Services (AWS) cloud.
ConPaaS [14]	This projects provides an open source system in which resources that belong to different operators are integrated into a single homogeneous federated cloud that users can access seamlessly. ConPaaS is a PaaS layer providing a set of elastic high-level services and runtime environments, including support for MapReduce jobs, Java and PHP applications, as well as as TaskFarming service. ConPasS is compatible with AWS and OpenNebula clouds.
HTCondor [10]	HTCondor is a workload management system for compute-intensive jobs. It provides a job management mechanism that automatically chooses when and where to run the jobs. In addition to being a *traditional* job manager, HTCondor scavenges CPU resources from potentially federated computers and makes those available via a single job queue. Standalone, HTCondor can also submit jobs to Amazon's EC2 resources.
Crossbow [15]	Crossbow is a domain-specific tool that focuses on whole genome resequencing analysis. It combines Bowtie and SoapSNP with Hadoop in a fixed data analysis software pipeline to enable fast data analysis on AWS or a local workstation.

III. IMPLEMENTATION

The CloudMan platform is currently available for immediate instantiation on the AWS public cloud and is also compatible with private or academic clouds based on OpenStack, OpenNebula, or Eucalyptus middleware (visit biocloudcentral.org). Deploying CloudMan on a private cloud requires creation of a custom machine image, configuration of which is automated via CloudBioLinux configuration scripts (see http://github.com/chapmanb/cloudbiolinux). An example of such a deployment is available on Australia's national cloud, NeCTAR, which is based on OpenStack middleware.

CloudMan itself is written in Python as a standalone web application and is open source licensed under the MIT license. Internally, CloudMan is based on service-oriented architecture; each of the components within CloudMan (e.g., file systems, job managers, applications) are defined as self-contained services. Services implement a standard interface and control all actions related to the given service. The master control thread manages and coordinates all of the running services. Instructions on how to use all of the features of CloudMan are available at (http://usecloudman.org) while the source code is available at (http://bitbucket.org/galaxy/cloudman).

A. Service Implementation: Hadoop

Along with the cloud resource management, resource allocation and management in a distributed system is a task, which requires a lot of effort. Various systems have been implemented to fulfill this need. Among these systems, Sun Grid Engine (SGE) has gained reputation for being a stable, efficient and reliable system. On the other hand, having the necessary infrastructure to ease the development of solutions for big data problems by using the computing power from distributed systems is quite demanding and Hadoop has been introduced as a solution for this task. Integrating Hadoop with SGE (i.e., providing an available and ready to use distributed architecture to the end user), gives us the capability of having the grid engine manage distributed resources while having Hadoop as a platform service available on demand. This approach was pioneered by [16], where Hadoop systems was pre-installed over cluster's nodes while the Sun Grid Engine allocated the necessary resources to its users on the fly. Although this solution is satisfactory for systems where cluster's nodes are pre-configured and Hadoop is the most demanding sub-system used, running Hadoop daemons over a cluster can be seen as waste of resource as it might not be in use over the entire lifespan of given cluster node. Furthermore, the available solution has separated Hadoop jobs from other jobs in the system, which would decrease job throughput by reducing the system's coherency.

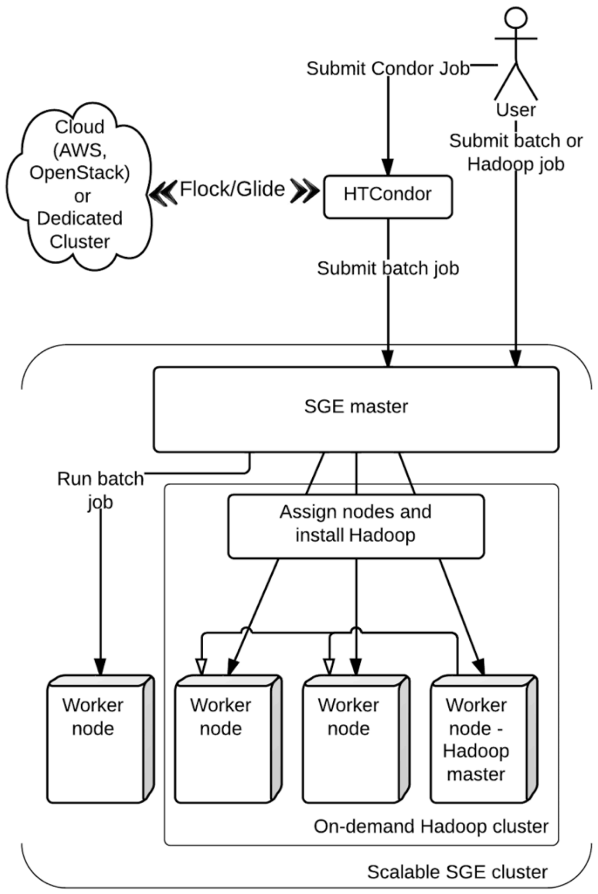


Figure 1. CloudMan architecture approach for supporting big data in a federated computing fashion.

We reengineered the available solution to increase system throughput by having Hadoop sub-system be provided by the grid engine on demand (vs. static). For this purpose, a grid engine integration script will inspect jobs submitted to SGE and whenever a Hadoop job is found, requested computing resources will be acquired from SGE and a Hadoop environment setup dynamically. In other words, before assigning a job to the resources, Hadoop will be installed on the acquired subset of the SGE cluster. The process of inspecting the job is realized in a constant amount of time, $O(1)$, and is handled by a job submission template script, which is made available to the end users. Each time a hadoop job is detected, the script will read the allocated resources from SGE and dispatch the required binaries to each resource. This process happens in parallel across all the nodes so the time complexity of deploying Hadoop over the allocated nodes is minimum and reduced to the amount of time required to dispatch Hadoop onto a single system. After Hadoop installation is completed, the job is sent to the Hadoop master node to run. Once the job completes, grid engine integration script cleans all the resources by removing any unnecessary files dispatched to each node and killing all the daemons. For a visual representation of this process please see figure 1.

B. Service implementation: HTCondor

Using private or community resources while having on-demand access to additional processing units from the cloud to spread the workload when necessary or desired is the promise that a hybrid IaaS solution offers. To achieve this goal, we adopted two concepts from HTCondor in our approach: flocking and gliding [17], [18]. In flocking scenarios, the work overflow will be sent to other, off-site resources to keep the runtime promises. Flocking was first introduced by the HTCondor team as a solution to submit the work overflow to other, remote resources. In this scenario, remote resources managed by a HTCondor component can be acquired when necessary, given they are not currently being utilized. Although flocking can fulfill off-site resource rental, it will not apply when remote resources are not managed by an HTCondor component. To solve this problem, gliding was introduced. The gliding scenario is a dynamic, on-demand deployment of HTCondor's necessary daemons over remote resources that are otherwise not controlled by an HTCondor component (i.e., daemon). In this solution, use of resources managed by a job manager (such as SGE or PBS) can be achieved. This is accomplished through a Globus Toolkit [19] connection string, which is shared between remote resources. The availability of joined remote resources can be restricted through the provided connection file. In the context of CloudMan, it is necessary for the remote organization resource to provide this string to CloudMan's local HTCondor. Although access will be granted through the connection file, gliding scenario will not be functional until the resource provider grants enough privilege to the CloudMan user to run necessary daemons over the remote resource. After having been granted sufficient privileges to run the daemons, the local HTCondor can send the job to the remote resource by receiving the remote resource IP or DNS and the path to the Globus file connection string.

Integrating these two concepts from HTCondor into our solution represents a first step in enabling CloudMan as a solution for hybrid IaaS. The actual integration with HTCondor is achieved by pre configuring the HTCondor settings in CloudMan and, whenever CloudMan starts, it sets up the HTCondor master node. As workers are added to the system, CloudMan automatically adds each worker node to the established condor pool, resulting in a single shared pool. To share the created resource pool in a flocking scenario, a negotiation between the two parties (resource provider and resource consumer) is required i.e. they should first come into a level of agreement on how, when and what resource can be used through the CloudMan. After reaching a level of agreement, the resource provider should grant sufficient privileges to CloudMan's HTCondor. This is achieved via CloudMan's DNS and HTCondor's configuration file. Lastly, the provider's DNS or IP should be passed into CloudMan, for which a form has been integrated into CloudMan's web interface. CloudMan will then set the necessary configuration, letting HTCondor negotiate and use remote cluster's resources as required (see Usage and Results section for an example).

IV. Usage and Results

Submitting a Hadoop job requires setting two parameters in a template script file. The script is provided by CloudMan and specifies the number of CPU slots the job requires and a path to the job executable. From here submitting this job is equivalent to submitting a any other SGE job:

```
$ qsub -v HADOOP_HOME=/home/ubuntu/hadoop/hadoop/home\
          ,JAVA_HOME=/usr <hadoop file script>
```

The challenge in this approach was to find a way to get both SGE and Hadoop to work without interfering with each other while keeping the resource utilization well-balanced. This goal is achieved by decoupling SGE from Hadoop where SGE prepares the necessary environment for Hadoop and then hands over the job to Hadoop. From this point on Hadoop handles the job as a self-contained system until the job has done. SGE is then informed of job completion and cleans the environment.

Figure 2 captures the process of creating the Hadoop environment within SGE and also shows sample times for the given process.

Submitting HTCondor jobs in a CloudMan cluster requires a condor job submission script to be created. Once composed, the script is submitted to HTCondor as shown below in figure 4.

Although employing HTCondor as an engine for the federated cloud resource management and gives us the capability of utilizing the disparate resources, the big data migration challenge remains an issue to be taken care of. Job files can be broadly grouped into two categories: the systematic ones, in which the system should take care of, and the user input data. In the context of CloudMan, a simple but effective solution for handling systematic data has been adopted; namely, any cluster managed by

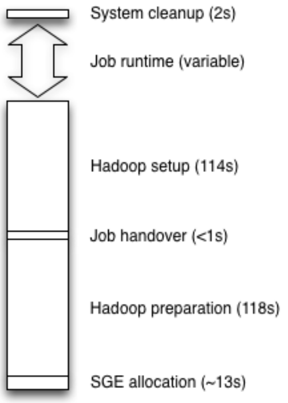
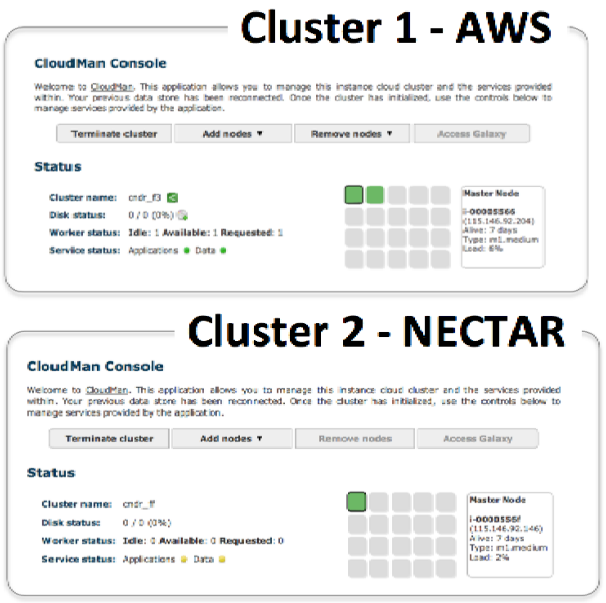


Figure 2. The Hadoop job preparation process through SGE and a sample timeline

CloudMan will have the same copies of files in their local directories. This includes installed applications, application configuration files, as well as reference data, which is often required by a variety of bioinformatics tools and reaches multiple terabytes. Therefore, jobs requiring access to these data can be used without any data copying from one CloudMan cluster to another. Whilst this is an efficient solution for systematic files, the user input data is a bottleneck in job dispatching. This is because flocking a job into a remote cloud or cluster requires transferring the input data next to the job, which means carrying the required amount of data over the network. Currently, CloudMan is relying on HTCondor job scheduler's decision for submitting jobs into the remote resources. We are exploring alternative scheduling approaches to utilize data locality between clusters with the aim of submitting data intensive tasks local to the data while sending compute intensive tasks to the available remote resources.



CloudMan Console

Welcome to CloudMan. This application allows you to manage this instance cloud cluster and the services provided within. Your previous data store has been reconnected. Once the cluster has initialized, use the controls below to manage services provided by the application.

Status

Cluster name: cndr_f3
Disk status: 0 / 0 (0%)
Worker status: Idle: 1 Available: 1 Requested: 1
Service status: Applications ● Data ●

Master Node
i-00085566
(115.146.92.204)
Alive: 7 days
Type: m1.medium
Load: 6%

CloudMan Console

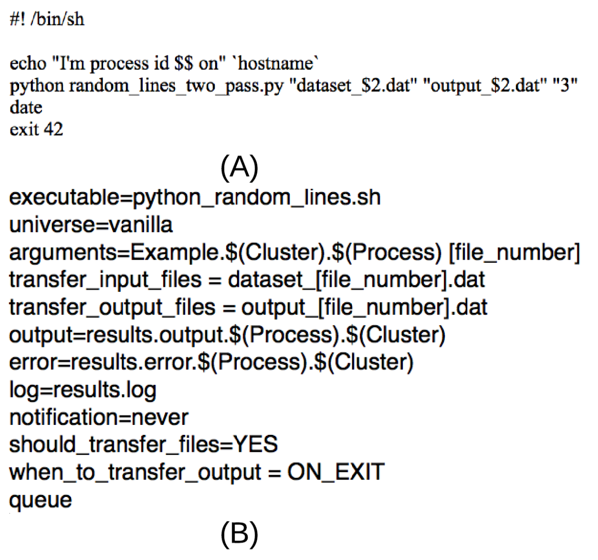
Welcome to CloudMan. This application allows you to manage this instance cloud cluster and the services provided within. Your previous data store has been reconnected. Once the cluster has initialized, use the controls below to manage services provided by the application.

Status

Cluster name: cndr_ff
Disk status: 0 / 0 (0%)
Worker status: Idle: 0 Available: 0 Requested: 0
Service status: Applications ● Data ●

Master Node
i-0008556f
(115.146.92.146)
Alive: 7 days
Type: m1.medium
Load: 2%

Figure 3. The CloudMan cluster setup for HTCondor flocking scenario. In this picture two CloudMan cluster has been created: one on the Amazon cloud (AWS) and one on the NeCTAR cloud (NeCTAR). The flocking happen from AWS to NeCTAR.



```
#! /bin/sh

echo "I'm process id $$ on" `hostname`
python random_lines_two_pass.py "dataset_$2.dat" "output_$2.dat" "3"
date
exit 42
```

(A)

```
executable=python_random_lines.sh
universe=vanilla
arguments=Example.$(Cluster).$(Process) [file_number]
transfer_input_files = dataset_[file_number].dat
transfer_output_files = output_[file_number].dat
output=results.output.$(Process).$(Cluster)
error=results.error.$(Process).$(Cluster)
log=results.log
notification=never
should_transfer_files=YES
when_to_transfer_output = ON_EXIT
queue
```

(B)

Figure 4. HTCondor flocking example. A is the script to be run by HTCondor in this exmaple it is named python_random_lines.sh. B is the HTCondor script which should be submitted to HTCondor named remote_job.submit. Here we submit these scripts multiple times into AWS cluster (From Figure 3) using multiple different input files. It is worth noting that the Exit 42 at the end of python_random_lines.sh will let HTCondor detect when the job is finished. Then afterwards it can transfer the related files back to the local HTCondor.

V. Conclusion

Infrastructure-as-a-service (IaaS) is the delivery of hardware and its associated software as a service [17]. CloudMan was first introduced to facilitate management of such a service with a focus on easing the process of preparing a basis for providing bioinformatics tools on the cloud. Successful projects, such as Galaxy, the Genomics Virtual Laboratory [20], CloudBioLinux, and SARA Cloud [21], are using or providing cloud services to the end users based on CloudMan.

In this paper we extended the capabilities of CloudMan to accommodate managing cloud resource in a more versatile and federated environment. This was achieved by integrating Hadoop platform into CloudMan and thus providing Hadoop-as-a-Service. With Hadoop's ability to process big data, availability of such a platform enables easier access to a new wave of (bioinformatics) tools for both, development and usage. In addition, we have integrated HTCondor into CloudMan. With HTCondor integrated, it is possible to easily span boundaries of a single compute cluster and extend the capacity of any administratively single resource, regardless of whether it is in the cloud or not. This enables overflowing excess workloads to a different set of computational resources (e.g., local cluster to a cloud or from one cloud provider to a different cloud provider). This also opens a new door towards federated job execution and data-locality based job execution.

Adding these features to CloudMan significantly extends the feature set of CloudMan and confirms its role as an accessible cloud management console and a provider of on-demand Platform-as-a-Service (PaaS) to the end users.

Acknowledgment

The CloudMan is being developed by the Galaxy (http://wiki.galaxyproject.org/GalaxyTeam) and LSCC teams (http://www.vlsci.org.au/page/lscc).

Funding: This work was, in part, supported by the Genomics Virtual Laboratory (GVL) grant from the National eResearch Collaboration Tools and Resources (NeCTAR).

References

[1] "The data deluge," *Nature Cell Biology,* vol. 14, August 2012.

[2] E. Afgan, D. Baker, N. Coraor, B. Chapman, A. Nekrutenko, and J. Taylor, "Galaxy CloudMan: delivering cloud compute clusters," *BMC Bioinformatics,* vol. 11, p. S4, 2010.

[3] E. Afgan, B. Chapman, and J. Taylor, "CloudMan as a platform for tool, data, and analysis distribution," *BMC Bioinformatics,* vol. 13, pp. 315-315, 2012.

[4] E. Afgan, D. Baker, N. Coraor, H. Goto, I. M. Paul, K. D. Makova, A. Nekrutenko, and J. Taylor, "Harnessing cloud computing with Galaxy Cloud," *Nature Biotechnology,* vol. 29, pp. 972-974, 2011.

[5] K. Krampis, T. Booth, B. Chapman, B. Tiwari, M. Bicak, D. Field, and K. E. Nelson, "Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community," *BMC Bioinformatics,* vol. 13, pp. 42-42, 2012.

[6] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome Biol,* vol. 11, p. R86, 2010.

[7] E. Afgan, J. Goecks, D. Baker, N. Coraor, T. G. Team, A. Nekrutenko, and J. Taylor, "Galaxy - a Gateway to Tools in e-Science," in *Guide to e-Science: Next Generation Scientific Research and Discovery*, K. Yang, Ed., ed: Springer 2011, pp. 145-177.

[8] A. Nekrutenko and J. Taylor, *Next-generation sequencing data interpretation: enhancing reproducibility and accessibility*.

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM,* vol. 51, pp. 107-113, 2008.

[10] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience," *CONCURRENCY AND COMPUTATION,* vol. 17, pp. 323-356, 2005.

[11] *High performance cluster computing. 1. Architectures and systems* vol. 1: Upper Saddle River, NJ [u.a.] : Prentice Hall, 1999, 1999.

[12] C. Ivica, J. T. Riley, and C. Shubert, "StarHPC — Teaching parallel programming within elastic compute cloud," *Proceedings of the ITI 2009*

31st International Conference on Information Technology Interfaces, p. 353, 2009.

[13] *Amazon Web Services*. Available: http://aws.amazon.com

[14] G. Pierre and C. Stratan, "ConPaaS: A Platform for Hosting Elastic Cloud Applications," *IEEE Internet Computing,* vol. 16, pp. 88-92, 2012.

[15] B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg, "Searching for SNPs with cloud computing," *Genome Biol,* vol. 10, p. R134, 2009.

[16] S. Krishnan, "myHadoop: Hadoop-on-Demand on Traditional HPC Resources," ed. The UC Cloud Summit 2011: University of California, Los Angeles, 2011.

[17] S. Bhardwaj, L. Jain, and S. Jain, "Cloud Computing: A Study of Infrastructure as a Service (IAAS)," *International Journal of Engineering and Information Technology,* vol. 2, p. 3, 2010.

[18] D. H. J. Epema, M. Livny, R. vanDantzig, X. Evers, and J. Pruyne, *A worldwide flock of Condors: Load sharing among workstation clusters*.

[19] I. Foster and C. Kesselman, "The Globus Toolkit," in *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds., ed San Francisco, California: Morgan Kaufmann, 1999, pp. 259--278.

[20] *Genomics Virtual Lab (GVL)*. Available: https://genome.edu.au

[21] M. d. Hollander, "The Cloud for Biologists," ed. SARA HPC Cloud Day, Amsterdam, 2011.