Interoperability of Commercial and Open-Source Clouds for Deployment of Bioinformatics Grid Computing Infrastructures with CloudBioLinux-CloudMan

Alex Richter[1], Konstantinos Krampis*[2], Enis Afgan*[3], Brad Chapman*[4]

[1] Informatics Department, J. Craig Venter Institute, 10355 Science Center Dr., San Diego, CA 92121, USA
[2] Informatics Department, J. Craig Venter Institute, 9704 Medical Center Dr., Rockville, MD 20850, USA
[3] Center for Informatics and Computing, Ruder Boskovic Institute, Zagreb, Croatia
[4] Dept. of Biostatistics, Harvard School of Public Health, 655 Huntington Avenue, Boston, MA, 02115, USA

Email: Konstantinos Krampis*- agbiotec@gmail.com; Enis Afgan*- afgane@gmail.com; Brad Chapman*- chapmanb@gmail.com;

*Corresponding author

Abstract

Background: Conversion of self-contained binary Virtual Machine (VM) files that encapsulate the operating system, bioinformatic tools and software libraries, is straightforward using utilities available by vendors of open-source and commercial clouds. Independent parallel processing of datasets with bioinformatics pipelines running separately within multiple instances of a specific VM is an efficient approach, based on results presented in the current study. Nonetheless, virtualization and replication within clouds of informatics infrastructures such grid computing, has significant merit given that a majority of current bioinformatics pipelines need significant code refactoring to be used outside of institutional grids.

Results: In the present study, we use as foundation the CloudMan framework that allows bootstrapping of virtualized Sun Grid Engine (SGE) infrastructures using VMs on the Amazon EC2 cloud. Within the specific context of Cloudman and its interactions with the Application Programming Interface (API) of the commercial Amazon EC2 or the open-source Eucalyptus and Openstack clouds for the purpose of instantiating virtualized grids, we first examine cross-compatibility across these clouds. Cloudman is build using components of web programming libraries that provide the interaction with the cloud API, and in our results we present special cases observed for each cloud we examined, while also argue on the feasibility for development of similar frameworks that allow virtualization of current institutional bioinformatics infrastructures and ease of deployment across the different

cloud platforms. Finally, we present some test cases on performance and efficient utilization of underlying physical server resources for data analysis pipelines utilizing this framework.

Conclusions: Unlike ground up development of web applications that are designed around based on the cloud APIs and to natively run within the cloud infrastructure, complex bioinformatics applications that are in advanced development state can seamlessly encapsulated within VMs on cloud platforms. Nonetheless, for taking advantage if the scalability offered by the cloud, frameworks like Cloudman that stand up grids or other commonly used approaches on institutional clusters are required, while also assuring that the framework interoperates across the different cloud platforms.

Background

Current State of Cloud Computing For Bioinformatics

In recent years, cloud based bioinformatics data analysis systems such as Galaxy [1], CloVR [2], Cloud BioLinux [3] and BioKepler [4] were released, allowing smaller laboratories and institutions to perform large-scale data analysis with genomic datasets. All these platforms are available on the Amazon Elastic Compute (EC2) cloud [5] , which provides data centers in US East and West regions, European Union, Asia and Australia [6], allowing any researcher worldwide to connect to the cloud end-point closest to their geographic boundaries.

The Amazon cloud allows users to run Virtual Machines (VM) with various capacities in terms of the underlying physical compute resources allocated to the VM, and the usage is billed on an hourly basis [7]. A VM is essentially a full-featured compute server, with virtual processors, memory and storage capacity, that runs on a Virtualization layer set on top of the hardware architecture on Amazon EC2. Furthermore, the operating system and all software components are encapsulated within the VM, and data analysis pipelines, databases, website portals and all their required code libraries can be distributed through a VM pre-configured and ready to execute. This approach for distributing software can remove many of the technical roadblocks for utilizing bioinformatics tools developed at various research institutions, and consequently make the software easily accessible to the research community.

Private clouds and Virtualbox where the VMs can run.

Cloud BioLinux and Fabric and CloudMan.

Open-Source Cloud Computing PlatForms As Private Clouds

Architecture of Eucalyptus Clouds and Installation at J. Craig Venter Institute

The Eucalyptus open-source cloud [8] offers a fully compatible Application Programming Interface (API) with Amazon's EC2 cloud, including the Simple Storage Service (S3) [9], Elastic Block Store (EBS) [10] and additional services such as Auto Scaling and CloudWatch. The Eucalyptus API providing access to the cloud services is installed and runs on a Cloud Controller (CLC) physical machine, and while Eucalyptus-specific tools are available for interacting with the API [11], the Amazon tools [12] are fully compatible according to our experience. The CLC is essentially a software layer that besides serving the API and allowing interaction with the cloud, also tracks at a high level the status of physical and virtualized computing resources, in addition to being the coordinating service for the other Eucalyptus layers that constitute the cloud running on the physical cluster. For large cloud installations, more than one CLC instances can be in place across different physical machines, ensuring scalability and load balancing when a large number of users interact with the cloud API, while also safeguarding the cloud operations in case of hardware failure.

In more detail, one of the cloud's software layers providing information to the CLC is the Cluster Controller (CC). The CLC delegates to the CC the task of gathering statistics from the physical nodes of the compute cluster running the Virtual Machines (VM), such as available disk space, number of VM running on each physical node, and compute load posed to the nodes. In large cloud installations, similar to the CLC multiple CC instances can be run on different physical machines for fail protection and to balance the load of keeping track of hundreds or even thousand of VMs concurrently being started, terminated and utilized at large capacity by users. The cloud installation at the J. Craig Venter Institute (JCVI) was comprised by four identical physical compute servers (16 core, 32GB memory, 1TB disk each), and one was used as the cloud's head node to run a single instances of the CLC and CC components.

Regarding the cloud components that the users can access storage and computational resources from, the Amazon S3 storage corresponding service on Eucalyptus is called Walrus. The Walrus and S3 storage model deviates from the typical file system of disk drives, it instead uses data objects that can be accessed by internet-identifiable unique Uniform Resource Location (URL) . A file with public permissions on a Walrus service for a cloud that is accessible outside of an institutional firewall, can be accessed by simply pasting

its URL on a web browser, but for upload, modification, or for bulk data operations or for files that require authentication, one of the two Application Programming Interfaces (API) must be used to access the service: the first is based on the Representational State Transfer protocol (REST, [13]) and the second on the Simple Object Access Protocol (SOAP, [14]). Alternatively, users can interact with the service through various desktop client applications that have graphical front-ends, available for all different operating systems [15–17].

Due to its front-facing role in serving data objects through identifiable URLs over the internet, Eucalyptus is designed to run the Walrus component from the same physical node of the CLC, which provide access to the cloud's API for the user's applications. Walrus can be from a virtual machine instance running inside the cloud, in the case a user application is accessing the data objects via their URLs. At JCVI's cloud, we run the Walrus service on the same physical compute server as CLC and CC, the simplest configuration available for Eucalyptus given the small size (four server nodes) of our cluster. According to recommendations on the Eucalyptus documentation [18], for larger cloud instances Walrus directory should be a mount of a Network Attached Storage (NAS, ref) system, allowing scalability for the amount of storage and performance in high user loads and data object requests.

Another role of the Walrus is to serve as the VM image store, where compressed splits of binary files composing the VM template are stored, copied and joined at the physical nodes that will run the VM during boot time. In our experience with the small Eucalyptus cloud where a NAS was not used but rather we run Walrus within the limitations of the a single disk drive on a physical server, copying binary splits of the VM template to the physical nodes during VM instance bootup, failed for VM larger than 100GB in size. We have found that using the caching mechanism available in Eucalyptus that places a VM template from Walrus on the CC [19] allowed us to boot VMs of size 200GB and up within a few minutes.

The Node Controller (NC) component of Eucalyptus, is the cloud layer that boots and runs the VMs on the physical compute servers of the cluster, and constantly provides information to the CC for the available compute resources and the number of VMs on the server. At JCVI we have configured our Eucalyptus cloud with 4 NCs and therefore we essentially provide a master physical server running all cloud components (CLC, CC, Walrus, NC), and 3 worker servers running a single NC each dedicated to controlling resources where the VMs are running.

The workflow among the different components discussed so far, followed within a Eucalyptus cloud setup to boot a VM is as following: using the command line tools available either by Eucalyptus or Amazon or any

of the available client applications, a call is made to the cloud API through the CLC for booting a VM with specified computational capacity; the CLC checks its registry for the availability of resources, and routes the request to a specific CC (if more than one CC are available), that has under its management a set of NC with available resources to run the VM with requested capacity; the CC checks whether the VM is cached locally and if not a copy is initiated from Walrus and passed over to the NC, while an unique IP address is assigned to the VM; finally, once the NC boots the VM its IP address and periodic updates of its status are communicated back through the same chain to the CLC and the user's application through the API.

The last component of Eucalyptus is the Storage Controller (SC), which corresponds to the Elastic Block Store (EBS) storage option on the Amazon cloud. This provides virtual disk drives with identical filesystems to physical drives (POSIX-compliant [20]), multiples of which can be attached to a running VM. While the Amazon EBS provides virtual disk drives up to 1 TeraByte (TB), on Eucalyptus the maximum size can be set through a configuration parameter and also depends on the space available on the underlying physical hard drive on the SC machine. While in theory the sizeof a single virtual data volume can reach up to the number of GB to fill the physical underlying hard drive, an upper limit has not been tested with the Eucalyptus cloud system. In our cloud and for our needs we started 2 250GB and a few smaller ones without any performance problems.

(find more from the other paper)EBS volumes persist past VM termination and are commonly used to store persistent data. An EBS volume cannot be shared between VMs and can only be accessed within the same availability zone in which the VM is running. Users can create snapshots from EBS volumes. Snapshots are stored in Walrus and made available across availability zones. Eucalyptus with SAN support lets you use your enterprise-grade SAN devices to host EBS storage within a Eucalyptus cloud. Snapshots are placed on Walrus providing backup and a template to create multiple copies of the EBS volume. On Amazon S3 since it is replicated across 3 zones, distaster recovery (look at the other paper)

These data volumes can be also used as the bootable file system of a running VM stored only on a single Amazon data center. The disadvantage is that the EBS is mounted from the SC to the NC, using only the memory and CPU of the physical servers running the NC, and creating a bottleneck the SC unless is connected to a NAS.

This is available but has not been tested in our Eucalyptus cloud. VMware Broker (Broker or VB) is an optional Eucalyptus component, which is available if you are a Eucalyptus subscriber. VMware Broker enables Eucalyptus to deploy virtual machines (VMs) on VMware infrastructure elements. VMware Broker mediates all interactions between the CC and VMware hypervisors (ESX/ESXi) either directly or through

VMware vCenter.

OpenStack

Enis and collaborators. Describe the OpenStack architecture and components (Nova, S3 etc) that it provides. Example private cloud installations that use it.

Results and Discussion

Porting the CloudBioLinux-CloudMan Framework to Private Clouds

The CloudMan Implementation For Scalable Computing on The Cloud

Also present CloudMan as an enabling framework for porting bioinformatics pipelines, with example pipelines and other applications implemented to run in parallel using the framework.

Enis, technical description of CloudMan as it stands for Amazon EC2. For example it uses two data volumes, pulls the code on boot from S3, how does it work with snapshots etc. This will serve as leeway for when describing how we mapped these components to the other platforms, see below.

Porting CloudBioLinux-CloudMan to Eucalyptus

Alex, technical description of changes to the CloudMan EC2 codebase to run on Eucalyptus. How components that CloudMan uses on EC2 where mapped to Eucalyptus. Specific nagging that Eucalyptus does with running CloudMan - i.e. different responses from the API returned when doing calls via boto, and changes in the cloudman codebase in order to take care of these. Also is there something that sysadmins that set up Eucalyptus should pay attention too (in the settings, in order to leverage those applications that create scriptable infrastructures by calling the API)

Porting CloudBioLinux-CloudMan to OpenStack

Enis, technical description of changes to the CloudMan EC2 codebase to run on OpenStack. Changes that have be made to code in order to communicate with the API correctly. What sysadmins that set up

OpenStack should pay attention too...

Execution and Performance of Bioinformatic Pipelines on Private Clouds

An HMM-BLAST Pipeline for Metagenomic Annotation at JCVI

Ntino... I got the numbers, need to write it.

A Variant Calling Analysis Pipeline at HSPH

Enis Other Example of Pipeline

Conclusions

...

Author's contributions

Alex Richter and Konstantinos Krampis worked on the technical aspects of porting CloudMan to the Eucalyptus Cloud. Enis Afgan and .... worked on porting CloudMan on the Openstack Cloud. Brad Chapman implemented the parallel sequence data analysis pipelines used for benchmarking the two different Clouds.

Acknowledgements

Text for this section ...

References

1. Goecks J, Nekrutenko A, Taylor J: **Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences**. *Genome Biol* 2010, **11**(8):R86.
2. Angiuoli S, Matalka M, Gussman A, Galens K, Vangala M, Riley D, Arze C, White J, White O, Fricke W: **CloVR: A virtual machine for automated and portable sequence analysis from the desktop using cloud computing**. *BMC Bioinformatics* 2011, **12**:356.
3. Krampis K, Booth T, Chapman B, Tiwari B, Bicak M, Field D, Nelson KE: **Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community**. *BMC Bioinformatics* 2012, **13**:42.
4. *Distributed workflow-driven analysis of large-scale biological data using biokepler*, ACM 2nd international workshop on Petascal data analytics 2011.
5. **Amazon Web Services Elastic Compute Cloud (EC2)** [htpp://aws.amazon.com/ec2].
6. **Amazon EC2 Cloud global regions** [http://aws.amazon.com/ec2/regions].
7. **Amazon EC2 Cloud Pricing** [http://aws.amazon.com/ec2/pricing/].
8. **Eucalyptus Open Source Cloud Platform**[http://open.eucalyptus.com].
9. **Amazon Simple Storage Service**[http://aws.amazon.com/s3].
10. **Amazon Elastic Block Store**[http://aws.amazon.com/ebs].
11. **Eucalyptus Cloud developer tools** [http://open.eucalyptus.com/wiki/Euca2oolsGuide].
12. **Amazon Cloud API tools**.
13. Fielding R: **Representational state transfer (REST)**. *Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine* 2000, :120.
14. **Simple Object Access Protocol (SOAP):**[http://www.w3.org/TR/soap].
15. **CuberDuck Amazon S3 cloud storage client**[http://cyberduck.ch/].
16. **S3browse** [http://s3browser.com/].
17. **Amazon S3 Firefox browser extension**[https://addons.mozilla.org/en-US/firefox/addon/amazon-s3-organizers3fox/].
18. **Eucalyptus Cloud Documentation**.
19. **Caching Virtual Machines on Eucalyptus**.
20. Mathur A, Cao M, Bhattacharya S, Dilger A, Tomas A, Vivier L: **The new ext4 filesystem: current status and future plans**. In *Proceedings of the Linux Symposium*, *Volume 2*, Citeseer 2007:21–33.