Reproducible Scientific Computing Environment with Overlay Cloud Architecture

Shigetoshi Yokoyama[*], Yoshinobu Masatani[*], Tazro Ohta[†], Osamu Ogasawara[‡], Nobukazu Yoshioka[*], Kai Liu[§], Kento Aida[*]

[*]National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, JAPAN
[†]Database Center for Life Science, Yata 1111, Mishima, Shizuoka, 411-8540, JAPAN
[‡]National Institute of Genetics, Yata 1111, Mishima, Shizuoka, 411-8540, JAPAN
[§]SOKENDAI (The Graduate University for Advanced Studies), Shonan Village, Hayama, Kanagawa 240-0193 JAPAN

***Abstract*— Science computing platforms are changing from traditional on-premises computing platforms to clouds. The style of research publications is also changing with the movement of open science. For example, it is common that researchers in the bioscience research community publish their research papers with the associated research data on the Internet. Furthermore, demands for reproducibility of computational experiments are increasing. A conventional method, or sharing source codes of application programs, is not enough to guarantee the reproducibility of computational experiments. In this paper, we propose overlay cloud architecture for building virtual clouds on cloud platforms. In the proposed architecture, the middleware, Virtual Cloud Provider (VCP), automatically configures computing environments (application programs, operating systems, libraries and binaries) that is required to reproduce computing results by using the Linux container technology. VCP also deploys containers of computing environments on clouds and configures overlay network connecting the deployed containers. A case study which considers DNA sequencing software shows how the VCP can be deployed, and used to reproduce results.**

Keywords—Cloud Computing; Container; Bioinformatics; Inter-Cloud;

I. INTRODUCTION

Cloud computing has been widely used for enterprise applications, and there is a growing interest in the academic community in the use of cloud computing for scientific applications. The performance of cloud platforms in scientific applications has been investigated in numerous studies [1][2][3]. Public cloud platforms, e.g. Amazon EC2 [4] and Microsoft Azure [5], have also been used to run scientific applications. For example, the 1000 Genomes Project uses Amazon Web Services for storing and analyzing genome data [6]. Now, science computing platforms are changing from traditional on-premises computing platforms to clouds.

The style of research publications is also changing with the movement of open science. Currently, researchers make their research papers publicly available in research repositories, e.g. institutional repositories, on the Internet. In some research communities, researchers publish their research papers with the associated research data on the Internet. For example, it is common for researchers in the bioinformatics research community to make the databases of these DNA sequences publically available, e.g. DDBJ, NCBI and EMBL-EBI [7], when they publish research papers. Furthermore, demands for reproducibility of computational experiments are increasing. The idea is enabling researchers in a research community to reproduce the computational results presented in original research papers on their computational platforms. The conventional method is sharing source codes of application programs, e.g. data analysis programs and simulation programs, which are used to produce the original computational results. Again, in the bio informatics research community, researcher runs a pipeline (or workflow), consisting of a chain of processing elements (or application programs), to determine the order of nucleotides in a DNA molecule [8] . A lot of source codes of the processing elements are publicly available.

However, the current efforts presented above are not enough to guarantee the reproducibility of computational experiments. First, sharing source codes are not enough to reproduce computational results. Computational results sometimes depend on not only application programs but also the underlying software, e.g. the operating system, libraries and binaries. In other words, we cannot guarantee the reproducibility between two computers with different underlying software, even when we build application programs from the same source code on both computers. Seconds, configuring a computing environment (application programs and the underlying software) for scientific applications is hard. Computing environments for scientific applications are becoming larger and more complex as problem sizes and data sizes are increasing. Currently, distributed computing platforms, or clusters of computers, are common for running data analyses or simulations in scientific applications. Furthermore, some environments may be built on computing resources distributed across geographically diverse locations and operated by independent cloud providers (or Inter-Cloud [9]). Reproducibly configuring computing environments on distributed resources requires expert knowledge and advanced technical skills about computer administrations, and it is not feasible for researchers in application domains.

2159-6190/16 $31.00 © 2016 IEEE
DOI 10.1109/CLOUD.2016.105

In this paper, we propose overlay cloud architecture for building virtual clouds on cloud platforms. In the proposed architecture, the middleware, Virtual Cloud Provider (VCP), automatically configures computing environments (or virtual clouds) on a cloud or Inter-Cloud platform(s). VCP solves two issues discussed in the previous paragraph as follows: first, VCP saves computing environments (including application programs, operating systems, libraries and binaries) that is required to reproduce computing results by using the Linux container technology. Second, VCP automatically deploys containers of computing environments on clouds and configures overlay network connecting the deployed containers.

VCP enables users to easily configure the customized computing environments on clouds. To this end, VCP automatically controls virtual machines (VMs) or bare-metal machines (BMs) through APIs on multiple cloud platforms, deploys containers of computing environments on VMs/BMs, and configures virtual networks among VMs/BMs. VCP is also able to configure customized computing environments on Inter-Cloud. In big science applications, data are produced and archived from many sources, e.g. experimental devices, sensors, databases, which are located around the world. It is not practical to collect and analyze all data on a single cloud platform. Inter-Cloud presents a solution to the limitations of a single cloud platform. We implemented the prototype of VCP and conducted experiments using DNA sequencing applications. We also demonstrate how VCP enables users to configure customized computing environments that can produce reproducible computing results. To the best of our knowledge, this is the first paper that demonstrates to automatically configure customized computing environments with reproducibility for use with DNA sequencing.

The remainder of this paper is organized as follows. Section II presents the background for this paper. Section III presents the proposed overlay cloud architecture and the implementation of VCP, and Section IV considers a case study of DNA sequencing. Finally, Section V summarizes the contributions of this paper and outlines future works.

II. Background and Related Work

In this section, we introduce underlying technologies and related work of the proposed overlay cloud architecture.

A. Linux Container

The Linux container technology is a lightweight virtualization technology of computing environments. Multiple containers running on the same host share the same operating system with the host, while coexisting virtual machines run their independent operating systems. Therefore, the container technology is able to provide near native performance, higher density and faster start/stop time. Modern Linux kernel features, such as Namespaces [24] and Cgroups[25], are building blocks for container technology. Namespaces are the key to isolate containers with the host and other containers, and Cgroups can help containers control the usage of resources like CPU and memory.

Docker[16] is the most popular and successful container management tool and supported by a lot of public cloud providers. Its great user friendliness contributes a lot for the success of Docker. The ability to package applications and its dependencies into a lightweight Docker image is another advanced feature of Docker. Therefore, users are able to download Docker images and run the application in any Docker enabled host without worry about laborious installation processes. This also helps users a lot to quickly deploy big data applications in clouds.

B. Distributed Computing Platforms with Containers

Distributed computing platforms, or clusters of computers, are currently common for running data analyses or simulations in scientific applications. A container cluster system helps users to build computing environments on distributed computing platforms with the Linux container technology. It controls multiple federated containers running on distributed computing resources, e.g. scheduling, deployment and execution. For application programs represented as workflow, the container cluster system is used as computing engine of the workflow. A lot of implementations are currently available, e.g. Mesos[18][27], Kubernetes[21], Docker Swarm [22]and HTCondor[23].

The workflow execution management system, Skyport[17], enables container-based workflows to run on multiple cloud platforms. The development of Skyport was based on the workflow management system, AWE/Shock. When a job is assigned to a computing resource (AWE worker), the AWE worker automatically downloads the input files and Docker container images needed to run the job. Skyport automates application deployment to multiple cloud platforms; however, it does not have a mechanism for configuring the virtual network among computing resources. Furthermore, Skyport assumes that the AWE/Shock framework is already installed on the cloud platforms. Thus, the user needs to manually install AWE/Shock if it is not available. It also imposes the limitation that only software which is compatible with Skyport can be used. In the proposed overlay cloud architecture, VCP automatically configures not only the computing resources but also the virtual networks between them. Moreover, VCP eliminates the dependency on preinstalled software, such as the workflow management system, by using the nested container mechanism described in Section III.A.

C. Inter-Cloud

Inter-Cloud is an architecture for organizing a cloud of clouds. It utilizes multiple cloud platforms to offer enhanced capabilities. Examples of Inter-Cloud being used for business applications have been discussed [8]. These include guaranteed end-to-end quality of service, enhanced convenience through service cooperation and service continuity. For scientific applications, Inter-Cloud can be used as a distributed computing platform to utilize computing, storage and network resources over the Internet.

Progress has been made in standardizing platforms (or providers) to allow them to interoperate with each other [12][13][14]. The standards usually concern models and protocols among cloud providers to enhance disaster recovery capabilities, service continuity and for cloudburst use. Testbeds for investigating inter-operation have also been

discussed [15]. Although such an approach offers numerous benefits, in reality very few implementations conform to these standards. This is because the implementation would have a huge initial cost and would require frequent upgrades of middleware on cloud platforms. The other approach is to use the overlay network for federating heterogeneous cloud platforms. Recent virtualization technologies, e.g. Docker and VPN technologies make this approach feasible. The proposed overlay cloud architecture adopts the latter approach.

Configuring computing environments in Inter-Cloud is not as easy as in a single cloud platform. The user needs to control virtual machines (VMs), deploy application software and their dependencies (binaries and libraries) on VMs, and configure virtual networks among VMs over multiple cloud platforms. Software is available to support the above configurations, e.g. RightScale [10]and OpenNaaS[11]. However, the existing software only partially supports these configurations, and the user still needs to perform time-consuming work to configure the customized computing environment in Inter-Cloud.

III. Proposed Architecture

In this section, we present the proposed overlay cloud architecture. The middleware, VCP, is the core technology for implementing the proposed architecture. We also present the design and implementation of the VCP.

A. Overlay Cloud Architecture

Fig. 1 presents an overview of the overlay cloud architecture. The overlay cloud architecture is defined by three layers, the real cloud providers' layer, the virtual clouds layer and the services (applications) layer, as illustrated in Fig.1. The real cloud providers' layer consists of cloud platforms operated by multiple cloud providers. The services layer is a collection of applications. The virtual clouds layer offers virtual clouds overlaid on virtual machines (VMs) or bare-metal machines (BMs), which are offered from real cloud providers, to applications. Here, the real cloud providers' layer is hidden from the services layer; that is, VCP configures a customized virtual cloud for each application where the application users neither have to manage VMs/BMs nor real cloud providers.

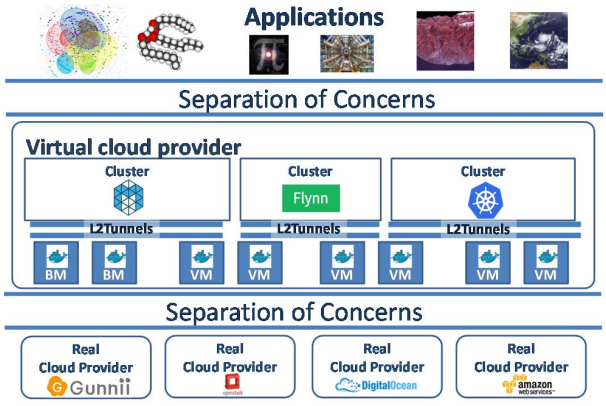


Fig.1. Overlay cloud Architecture

The primary purposes of the VCP are the deployment of software, including applications and the underlying system software, and the creation of a virtual network connecting VMs/BMs over multiple cloud platforms. To this end, VCP uses the container technology, Docker, to deploy software and connects the containers via overlay networks over virtual networks provided by the real cloud providers. We can assume that the overlay cloud architecture works on the current cloud platforms in the real world. Most public cloud providers currently support Docker engines and the user is able to run the same container images on different resources in the different cloud providers. In addition, we assume that the overlay networks can be created using Linux standard network services like Open vSwitch. Under these assumptions, the user is able to deploy containers on VMs/BMs over real cloud providers without any prior negotiations with the providers.

The software deployment process consists of two phases: the base containers deployment and application containers deployment phases. We assume that the user runs a workflow of an application on distributed computing resources. It is common that the workflow application is executed using cluster management software, e.g. a job scheduler, in the distributed computing systems. In the base containers deployment phase, containers with cluster management software are deployed on VMs/BMs organizing a virtual cloud. We refer to these containers as base containers. In the application containers deployment phase, containers with application programs, or application containers, are deployed on the VMs/BMs. Currently, some cluster management software, e.g. Mesos [18][27], allows containers to be run in the managed cluster. The application containers are executed with the cluster management software in the base containers. The overlay cloud architecture utilizes nested containers (a container within a container). Performance issues arise depending on the configuration of the nested virtualization layers. Our preliminary experiments show that the performance degradation due to the use of nested containers is negligible. The performance comparison among native BM, docker on BM and docker in docker on BM by UnixBench [28] is shown in Fig.2. The software and hardware configuration are summarized in Table.1.

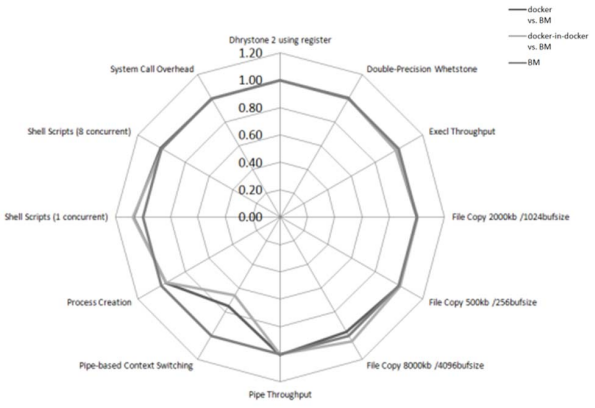


Fig.2. Bemchmark Results of Container in Container Setting

Table 1. The Software and Hardware Configuration of the Evaluation

OS	Ubuntu 13.10 (64bit)
Kernel	3.11.0-19-generic
Docker	0.10.0
OpenStack	2013.2 (Havana)
UnixBench	5.1.3
libc6	2.17-93ubuntu4
dash	0.5.7-3ubuntu1
coreutils	8.20-3ubuntu5
grep	2.14-3
gcc	4:4.8.1-2ubuntu3

--storage-driver="devicemapper"
Intel Xeon E5-2670 2.60GHz
32cpu (2socket x 8core x 2thread)
RAM 96 GB

The results show no significant performance degradation in the container in container configuration except "Pipe-based Context Switching". This benchmark spawns a child process and measures the context switching performance between two processes with pipe. Thus, we can assume that it increases the load of virtualization layer and we see the performance degradation in the docker on BM configuration. However, we also observe the slight performance degradation in the docker in docker configuration compared with the docker setting.

The deployment of virtual networks over VMs/BMs in Inter-Cloud depends on the underlying network architecture. In the typical scenario where the user runs applications on virtual machines in public clouds, VCP creates a virtual network between resources using conventional technology, e.g. IPsec. However, the communication performance may be highly degraded in this scenario, if secure communications are used. The other scenario is that the user runs applications on bare-metal machines (or bare-metal clouds) that are directly connected to the academic network with a high-speed VPN service, e.g. SINET L2VPLS[19]. We can achieve high-performance and secure communication between computer resources by using a high-speed VPN service.

B. Virtual Cloud Provider

VCP is the core technology for implementing overlay cloud architecture. Here, we describe how the VCP meets the requirements of applications running in the overlay cloud architecture. Below are requirements for VCP in the proposed overlay cloud architecture:

Req-0) Users are able to build a customized computing environment, or a virtual cloud, for the user applications.
Req-1) Users are able to use resources on major cloud platforms in the virtual cloud on demand.
Req-2) No prior negotiations are needed among cloud providers to build the virtual cloud other than becoming a regular cloud user.
Req-3) Users are able to use the full capabilities of the physical resources for running the user application.
Req-4) The virtual cloud should be scalable to the size of the problem.
Req-5) The virtual cloud should support applications from various scientific fields.
Req-6) Users are able to migrate applications to computing resources or sites, where huge databases or supercomputers are operated, in order to minimize data transfer time on the wide area networks.
Req-7) Users are able to run applications on confidential data in a secure manner (on Inter-Cloud).

We designed the VCP with the following criteria to fulfill the above requirements:

1) We improve the portability of applications by creating containers for applications and deploying these containers to resources in the virtual cloud. This fulfills Req-0, Req-1, Req-2, Req-3 and Req-5.
2) We improve the scalability of the virtual cloud by creating containers for cluster management software (base containers) and deploying the base containers to distributed resources in the virtual cloud. This fulfills Req-0, Req-4 and Req-6.
3) We connected base containers with a high-speed, secure VPN service in an academic network, e.g. SINET VPLS. This fulfills Req-3 and Req-7.

According to the criteria, the proposed VCP architecture consists of application container management, base container management, network management and container image management. In order to keep virtual cloud portability among various real cloud platforms, VCP has to handle those cloud managements APIs to obtain computing resources for running base containers on which application containers are able to run. This container in container configuration is needed to improve the portability of applications and virtual clouds.

The network management consists of two layers, the application container networking and the base container level networking. The network connecting base containers should be established with the layer 2 (L2) connections, so that the network configuration for application containers is more flexible. The connection among base containers is realized by the connection among host computers which are allocated from various cloud providers. From network performance points of view, it is better to use VPN services offered from wide area network providers than using L2 over L3 tunnels among the hosts.

Our preliminary experiments show that the network performance of the tunnel with IPsec for security is not practical for some applications. From this regard, we chose L2VPN for the network for base containers and L2/L3 tunnels for application containers. The application containers network

can be implemented on the L2VPN network among base containers. We are also able to implement it with VXLAN tunnels because it is becoming popular in container network implementation in the communities like the Docker eco-system.

C. Implementation of the VCP

We developed a prototype of VCP, which provides four services:

1) Cloud Resource Management

The cloud resource management service allocates computing resources (VMs or BMs) in real cloud providers to base containers. The user is able to modify the resource configuration (adding/deleting resources) on demand.

2) Base Container Management

The base container management service determines suitable computing resources for the deployment of base containers and configures parameters when launching containers. It also configures network settings, upgrades base containers or deletes containers in the same cluster at once.

3) Network Management

The network management service configures virtual networks, or VLANs, that connect base containers. It also configures the gateway of VLANs.

4) Base Container Image Management

The base container image management service consists of two repositories. The container image repository archives images of base/application containers, and the template repository archives templates of application configurations.

The VCP uses an orchestration mechanism, which automates deployment, configuration and management of the virtual cloud for the user applications. We use open source software, Terraform[26], for this orchestration in the VCP. Terraform has the following advantages.

- Terraform has a plug-in mechanism that enables the user to add third-party services.
- Terraform enables the user to configure multiple services in a single configuration file.
- Terraform enables the user to manage not only IaaS resources but also PaaS/SaaS resources.

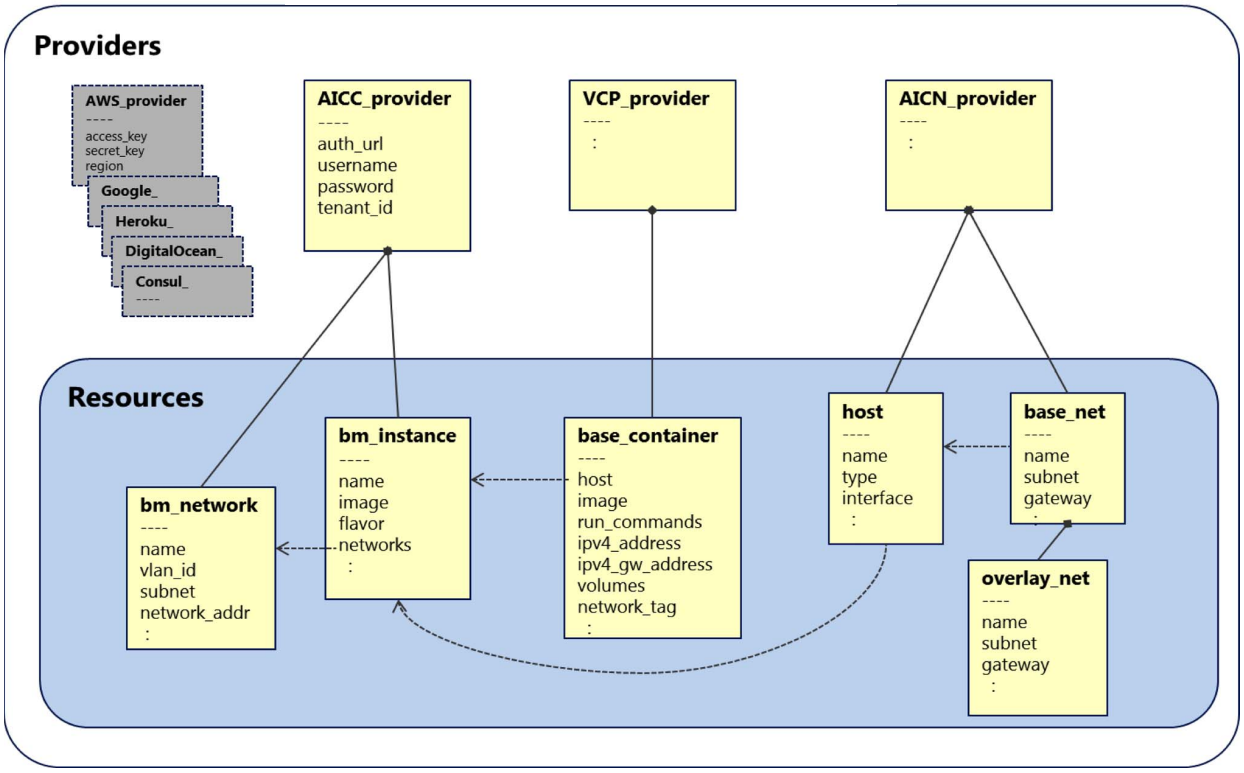


Fig.3. Plug-in Services Available in Terraform

We implemented the following three services using the plug-in mechanism of Terraform. Fig. 3 shows the relationships between the services.

- AICC (Academic Inter Cloud Computing) Provider

  The AICC provider is an implementation of the cloud resource management service customized for the cloud platform (AIC) operated in our institute, National Institute of Informatics. The AIC is an IaaS offering BMs. The AICC provider enables the user to control computing resources in the AIC. The AIC is operated using OpenStack. The APIs for controlling computing resources in the AICC provider are compatible with those of OpenStack. The proposed architecture enables to add providers for other cloud platforms, e.g. AWS, as illustrated in Fig. 3.

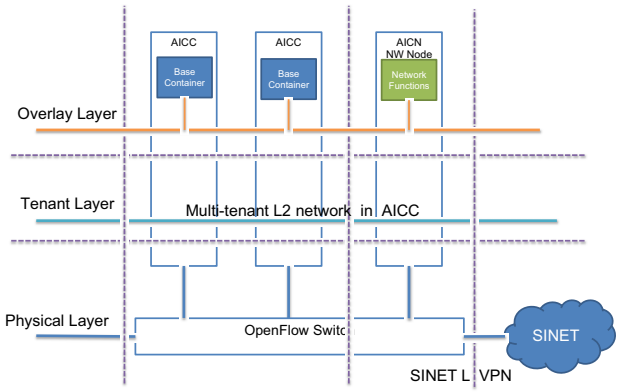


Fig.4. Network layers in virtual cloud

- VCP Provider

  The VCP provider is an implementation of the base container management service. It enables the user to control base containers in the Terraform framework.

- AICN (Academic Inter Cloud Network) Provider

  The AICN provider is an implementation of the network management service. A network in a VCP's virtual cloud consists of multiple layers, as illustrated in Fig. 4. The overlay layer connects base containers, and the tenant layer connects VMs/BMs in a cloud platform. The AICN provider controls both overlay and tenant layers.

As shown in Fig. 3, the AICC provider calls AICC APIs in order to prepare BMs for base containers. The AICN provider configures base networks, which connect the BMs using AICN APIs. It also creates overlay networks in the base networks to connect base containers, that is, it makes clusters of base containers. Then, the VCP provider deploys base containers on the clusters. Solid lines show relationships between plug-in services and resources and dotted lines show links among resources.

IV. Case Study of Bioinformatics Application

We conducted a preliminary experiment to build a virtual cloud for DNA sequencing applications using the VCP. Fig. 5 shows the customized computing environment for DNA sequencing in the experiments. The researcher runs a workflow of multiple software tools to determine the order of nucleotides in a DNA molecule. Galaxy[8] is a widely used web-based platform for managing workflows for DNA sequencing (Fig. 5). It enables the user to import software tools from the workflow repositories, create a workflow and submit the workflow to computing resources. The computing resources are operated with the cluster management software as outlined in Fig. 5. We use Mesos/Aurora cluster management software in this experiment. Thus, jobs in the workflow submitted from Galaxy are executed through the Mesos/Aurora framework.

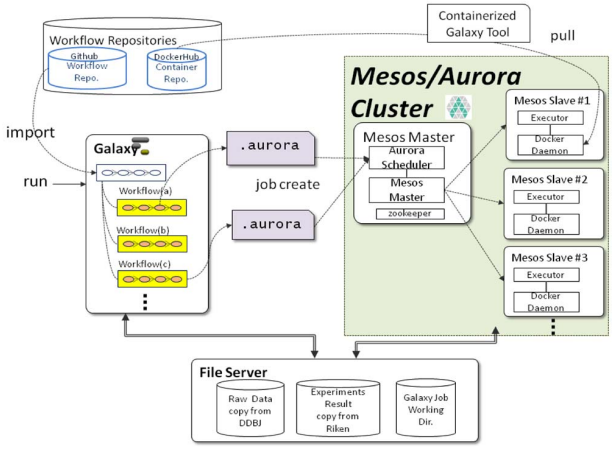


Fig.5. Computing environment for DNA sequencing application

In software deployment in the VCP, we created base containers for the Mesos/Aurora framework and Galaxy. The VCP deploys the base containers to computing resources in the cloud platforms, AIC. The file server in Fig. 5 allows application containers to access data in the server. Applications could be deployed to distributed computing resources in the Internet; thus, the file servers in the figure would be able to access distributed computing resources. There exist several options for the configuration of the file server, e.g. distributed file system, object storage or a network pipe. We used NFS for the file servers in the experiments. However, we leave the experiment with other implementations for future work.

Fig. 6 shows a workflow of the DNA sequencing application, RNA-Seq[20], which is used for analyzing the sequence of RNA in a cell. In the experiment, we created application containers for jobs in the workflow, "TopHat2", "Cufflinks", "Cuffmerge" and "Cuffdiff". Then, we deployed base containers, the Mesos/Aurora framework and Galaxy, to four computing nodes (Intel Xeon E5-2670 2.60 GHz 8-core x2CPU, 96 GB memory) in AIC. Here, the Mesos master, the Aurora scheduler, Galaxy and file servers are deployed to the single node. The Mesos slaves are deployed to the other three nodes. The application containers are deployed in the

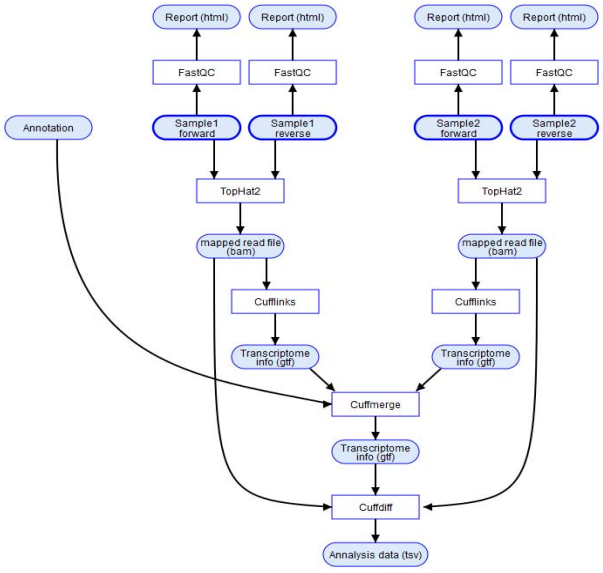


Fig.6. Workflow of RNA-Seq application

Mesos/Aurora framework and the jobs are scheduled and executed on the Mesos slaves.

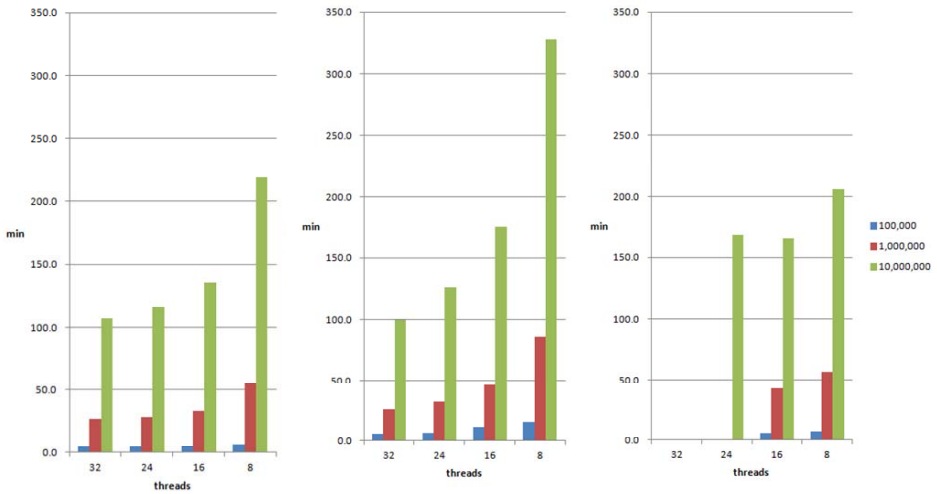
We confirmed that the VCP repeatedly deploys the above computing environment. The results indicate that the VCP enables the user to deploy the customized computing environment and that the results are reproducible. In order to confirm that VCP is able to deploy application framework on inter-cloud environment and run application containers on it easily, we deployed this reproducible environment in three real cloud providers configuration. One is AIC only, the second is AIC and the Hokkaido University cloud and the other are on Amazon Web Services.

The application can be deployed by VCP easily and run on each environment in reasonable performance as shown in Fig. 7. Most importantly, the outputs of the tasks in the workflow are equal in them. Each graph shows execution time of the workflow in three input data sizes. There are three inter-cloud experimentation configurations which are described in the bottom of the figure. The jobs are parallelized in Mesos slaves dividing the input files equally.

Fig. 8 shows the execution time for RNA-Seq on the above computing environment in AIC . The size of the input file, "FastQC" in the workflow, is 12 GB. We parallelized the longest job, "TopHat2", in the workflow to 24 parallels, because each sub-job is set to run 4 thread and 96 cores are allocated in this experiment, which resulted in an increase in the processing speed. However, the improvement in the performance is saturated at 8 parallels due to the performance bottleneck in the NFS servers.

V. Conclusions

We proposed the overlay cloud architecture for building virtual clouds on cloud platforms. In the proposed architecture, the middleware, Virtual Cloud Provider (VCP), automatically configures computing environments that is required to reproduce computing results by using the Linux container technology. VCP also deploys containers of computing environments on clouds and configures overlay network connecting the deployed containers. We developed



Mesos Master, Galaxy@AIC
Mesos Slave #1@AIC
Mesos Slave #2@AIC
Mesos Slave #3@AIC
Mesos Slave #4@AIC
AICN Node (NAT Gateway) @AIC
NFS Server (/mnt1/nig, riken) @AIC
Docker Hub

Mesos Slave Spec
Intel Xeon E5-2670 2.60GHz
32cpu (2socket x 8core x 2thread)
RAM 96GB

Mesos Master, Galaxy@AIC
Mesos Slave #1 @hokudai
Mesos Slave #2 @hokudai
AICN Node (NAT Gateway) @AIC
NFS Server (/mnt1/nig, riken) @AIC
Docker Hub

Mesos Slave Spec
Intel Xeon E7-8870 2.40GHz
40cpu (4socket x 10core)
RAM 128GB

Mesos Master, Galaxy@AIC
Mesos Slave #1 @AWS
AICN Node (NAT Gateway) @AIC
NFS Server (/mnt1/nig, riken) @AIC
Docker Hub

Mesos Slave Spec
Intel Xeon E5-2666 v3 2.90GHz
16cpu (1socket x 8core x 2thread)
RAM 30GB
(c4.4xlarge)

Fig.7. Execution Time of the Workflow on Three Cloud Environments

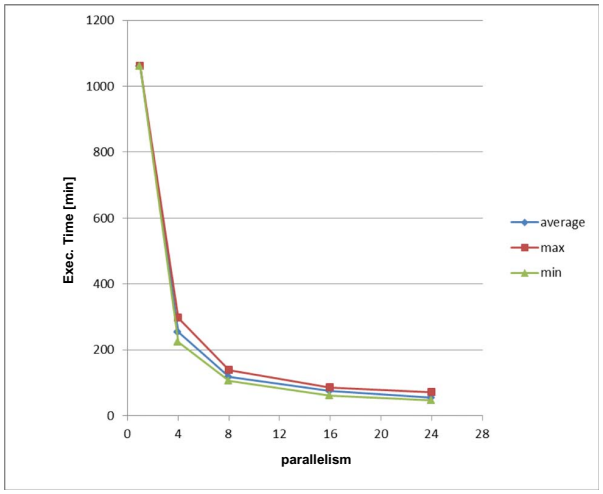


Fig.8. Execution Time of TopHats in RNA-Seq Workflow

the middleware, VCP, and presented a case study involving a DNA sequencing scenario. The preliminary experimental results show that the VCP enables the user to deploy the computing environment that produces reproducible results.

Our future work includes case studies of VCP in Inter-Cloud. The case study presented in this paper is obtained from preliminary experiments on small settings. We need to conduct more experiments with different settings with different cluster management software, e.g. Kubernetes[21]. Experiments on larger settings with multiple clouds are also an area for future work.

In the preliminary experiments, the application framework schedulers determine how resources are used by the application containers, and application providers determine the base container allocation. In the inter-cloud situation, the allocation of resources has to be done dynamically for ease of use, resource efficiency and application performance. The design and implementation of the mechanism is our future work. There are further areas for consideration, such as how to select the optimal virtual network deployment process, which we leave for future work.

Acknowledgment

We thank Mr. Jun Nasuno for his support in the implementation of the VCP and the experiments. This work was supported in part by CREST, JST.

References

[1] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B.P. Berman, and P. Maechling, “Scientific workflow applications on Amazon EC2,” Proceedings of the 2009 5th IEEE International Conference on E-Science Workshops, pp.59–66, 2009.

[2] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H.J. Wasserman, and N. Wright, “Performance analysis of high performance computing applications on the Amazon Web Services cloud,” Proceedings of the 2010 IEEE 2nd International Conference on Cloud Computing Technology and Science (CloudCom 2012), pp.159–168, 2010.

[3] A. Marathe, R. Harris, D.K. Lowenthal, B.R. de Supinski, B. Rountree, M. Schulz, and X. Yuan, “A comparative study of high-performance computing on the cloud,” Proceedings of the 22nd international symposium on High-performance parallel and distributed computing (HPDC ’13), pp.239–250, 2013.

[4] Amazon Web Service , http://aws.amazon.com/ (accessed on 2/22/2016)

[5] Microsoft Azure, https://azure.microsoft.com/ (accessed on 2/22/2016)

[6] 1000 Genomes, A Deep Catalog of Human Genetic Variation, http://www.1000genomes.org/

[7] International Nucleotide Sequence Database Collaboration, http://www.insdc.org/

[8] Galaxy, Data intensive biology for everyone, https://galaxyproject.org/ (accessed on 2/22/2016)

[9] Global Inter-Cloud Technology Forum, “Use Cases and Functional Requirements for Inter-Cloud Computing,” Technical Report, Global Inter-Cloud Technology Forum, 2010.

[10] RightScale, http://www.rightscale.com/ (accessed on 2/22/2016)

[11] J.I. Aznar, M. Jara, A. Rosello, D. Wilson, S. Figuerola, “OpenNaaS based management solution for inter-data centers connectivity”, IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013), vol. 2, pp.45-50, 2013.

[12] Open Cloud Computing Interface (OCCI), http://occi-wg.org/ (accessed on 2/22/2016)

[13] Open Grid Forum, Network Service Interface (NSI), https://redmine.ogf.org/projects/nsi-wg/

[14] Distributed Management Task Force, Inc. (DMTF), Open Virtualization Format (OVF), http://www.dmtf.org/standards/ovf/ (accessed on 2/22/2016)

[15] D. Bernstein, Y. Demchenko, "The IEEE Intercloud testbed - Creating the global cloud of clouds," Proceedings of the 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013), vol.2, pp.45-50, 2013.

[16] Docker, https://www.docker.com/ (accessed on 2/22/2016)

[17] W. Gerlach, W. Tang, K. Keegan, T. Harrison, A. Wilke, J. Bischof, M. D'Souza, S. Devoid, D. Murphy-Olson, N. Desai, F. Meyer, "Skyport - container-based execution environment management for multi-cloud scientific workflows," Proceedings of the 5th International Workshop on Data-Intensive Computing in the Clouds (DataCloud), pp.25-32, 2014.

[18] The Apache Software Foundation, Apache Mesos, http://mesos.apache.org/ (accessed on 2/22/2016)

[19] S. Urushidani, S. Abe, K. Yamanaka, K. Aida, S. Yokoyama, H. Yamada, M. Nakamura, K. Fukuda, M. Koibuchi, and S. Yamada , “New directions for a Japanese academic backbone network”, IEICE TRANSACTIONS on Information and Systems, vol. 98, no. 3, pp.546-556, 2015.

[20] Pitagora-Galaxy, Workflow RNA-seq 01, http://wiki.pitagora-galaxy.org/wiki/index.php/Workflow_RNA-seq_01 (accessed on 2/22/2016)

[21] Google, Kubernetes, http://kubernetes.io/ (accessed on 2/22/2016)

[22] Docer Swarm: a Docker-native clustering system https://docs.docker.com/swarm/ (accessed on 2/22/2016)

[23] HTCondor, https://research.cs.wisc.edu/htcondor/ (accessed on 2/22/2016)

[24] Namespaces. https://lwn.net/Articles/531114/ (accessed on 2/22/2016)

[25] Cgroups. https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt (accessed on 2/22/2016)

[26] Terraform, https://www.terraform.io/ (accessed on 2/22/2016)

[27] B. Hindman, A. Konwinski, M. Zaharia et al., “Mesos: A Platform for Fine-grained Resource Sharing in the Data Center,” in Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, Mar. 2011.

[28] UnixBench, https://code.google.com/archive/p/byte-unixbench/ (accessed on 2/22/2016)