Design and Implementation of the Application-level Migration Tool on Multi-site Cloud

Duy Nguyen, Nam Thoai
Faculty of Computer Science and Engineering
HCMC University of Technology
Ho Chi Minh city, Vietnam
nam@cse.hcmut.edu.vn

Abstract—Cloud computing is a powerful technique to increase hardware utilization dramatically and to scale up computational infrastructure instantly. To improve resource utilization, the cloud resource management system could suspend/migrate some running jobs/virtual machines in order to reserve resources for other ones. While the cost of moving the whole virtual machines is too high due to the big size of virtual machines, application level migration is a possible solution. We are interested in using checkpointing and recovery technique (C&R) to stop and restart running programs on virtual machines. The paper introduces the event-based checkpointing tool (EBC), in which users can easily checkpoint and restart running programs in cloud systems. EBC is based on event-driven architecture and supports both sequential and parallel programs like MPI programs. EBC is a useful tool supporting migration as well as scheduling in cloud systems.

Index Terms—**cloud, migration, checkpoint/restart, event-based system**

I. INTRODUCTION

Infrastructure services (Infrastructure-as-a-service), provided by cloud vendors allows any user to provision provide infrastructure on demand by utilizing the virtual resources to perform data/compute intensive works. By using IaaS, consumers do not need to investigate and maintain groups of machines and resolve the issue of using them efficiently. Instead, they simply find a suitable IaaS that provides the infrastructure having the desired (virtual) hardware configuration and corresponding built-in service/application. With recent development in cloud computing, it has become easier to build up a system that provides bigger computational capability at lower costs on clouds.

While there have been existed in the market a number of IaaSes, the benefits among providers are different. For example, Amazon SpotInstance provides virtual machines (VMs) base on the auction mechanism; Rackspace service uses pay-as-you-go mechanism; BitRefinery prices service usage at monthly period; GoGrid provides both mechanism. From the point of customer view, the computing environment in cloud can be scaled easily by starting/stopping multiple VMs. The customers can optimize their utilization by combining different provider's benefits. Despite the difference, VMs from different IaaS providers, called sites, share the same software and operating system as customer requirement. Applications can be migrated from one cloud provider to another to achieve the resource utilization.

In this paper, we investigate mechanism can be used to achieve the goal of deploying parallel application on multiple discontinuous cloud sites while reducing the cost of migration. While migrating whole VMs, our solution try migrate only images of application by applying the checkpointing & recovery (C&R) technique supported in OpenMPI [5]. Moreover, we propose an event-based checkpointing tool that can adapt to current middleware and benefit from auction-based in pricing. Such a tool is the Event-Based Checkpointing tool named EBC. The EBC architecture has been introduced in [9]. In this paper, we introduce details of the EBC design as well as implementation.

The rest of this paper is organized as follows. Section II introduces C&R technique. In section III, we discuss the computing environment on multi-site cloud and some of challenges for application-level migration. We describe our EBC system architecture in section IV and the implementation in section V. Section VI evaluates the performance of our technique for MPI applications. Section VII concludes the paper.

II. CHECKPOINTING & RECOVERY

Checkpointing is a technique to reduce the amount of lost computation by saving processes' states during failure-free execution and restarting from the saved states later [4]. To checkpoint/restart a process, all the components, which make up the processs state, must be considered. This set of data is also called the processes' image, which includes several segments, e.g. text, data (containing global variables and heap), stack, and other miscellaneous state information about the process, which is maintained by the kernel. In order to minimize the size of the stored process image, the amount of information required for checkpoint/ restart should be minimal. (For example, it is not necessary to save the text segment.)

Currently, many MPI programs use Open MPI [10], which has supported fault tolerance since its version 1.3 and this work is continuous. Open MPI asks an external checkpoint/restart system like BLCR [3] to save state of each process. It allows users using command lines to checkpoint/restart MPI programs. The time is wasted to checkpoint applications with number of processes from 5 to 100 showed in Fig. 1.

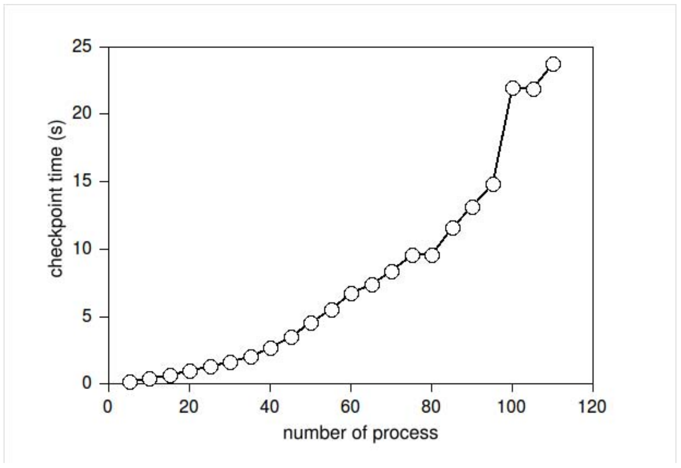

Fig. 1. Checkpointing time

III. Cloud environment on multi-site and migration

A. Multi-site Cloud

Cloud environment can be build for private purpose and in spare time it could be sharing between organizations. This toward the scalability issue since each private cloud has its own management system. To run a huge job which size is out of cloud border, it requires extra computing nodes from other sites includes access privilege, network bandwidth or storage space. In this sharing context, a site manager considers its resource as a solid zone and it must applying policy to each access from the outside. Due to the management separation, the user running that task does not know much information of the allocated nodes.

A site joining/leaving system at any time made it hard to keep track the structure of system network. Moreover, a remote site can be created by nodes placed at different places. The current network is based on location-oriented architecture so it does not allow to access an address outside of its container. Communication among different sites has a significant challenge of location-oriented dependencies due to the complexity of managing changes in the node's geographic and topological location. From point of routing view, the networking must provide a mechanism to deliver data to a group and all of its members. Various solutions have been discussed to enable multisite communication using ViNe router [7] and multicast; however, these communication solutions provide all-to-all communication but they still rely on IP address and require information of network structure [1].

B. Application Level Migration on Multi-site Cloud

A job expands its execution on other computing nodes migration from the source to target node. The migration at process level is difficult to use for real-world application due to 'residual dependencies' issue [8]. Application migration contains one of more process share the same issues with process migration. Although migration of the whole virtual machine has been proposed to avoid the problem of residual dependencies [2], its overhead and downtime is concerned due to the huge size of virtual machine. Live migration is another trend to minimize the downtime during migration but it still lack of the image size. While application migration takes images of running processes and then transfer these images to the target to recover the processes. The embedded checkpoint mechanism helps to overcome issues of process migration while it still keeps the small size of application image.

In virtualized environments, a network resource can be re-used for multiple networks or multiple resources can be aggregated for virtual resource. However, to manage these virtual resources effectively we need a management system. In cloud computing, the VM can be started on demand and stop when no longer to be used. The life cycle of VM is dynamically changed that we cannot use the static address. The creation of a new VM is different from action when a machine is configured before join/leave system. The cloud network is based on IP and various IP transport protocol. IP address includes both locator and identification. The IP address is global unique make some limitation. It is not suitable to deploy on multi-site cloud with requirement of scalability and mobility. In multi-site cloud, each cloud is a group of virtual machines managed by a frontend. The frontend is responsible for creating virtual machine and storing the related information. The stored data include VMID (identification of VM) but it is lack of the IP address because of VMM limitation. The addressing problem exists not only inside a cloud but also in case one VM need to communicate with other VM in different cloud. In cloud manager view, it can known VMID but not IP address.

IV. EBC: The event-based checkpointing tool

EBC [9] is a conceptual tool support migrating application over multi-site cloud using the application checkpoint capability. It also provides all-to-all communication toward address agnostic and node initiator connection. EBC provides mechanism to communicate between VMs even the VM information has not been collected by portal or multi-site cloud manager. This communication system helps overcome the limitation of IP address and allow directly connection from outside to a VM in a site. EBC system includes separated entities: nodes and brokers. These entities are started at its related cloud infrastructure element. During system construction, site join/leave, the connection among EBC entities is established to create the communication subsystem.

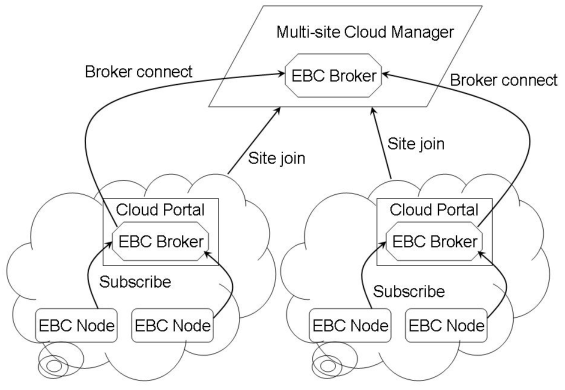

Fig. 2. EBC system

As in Fig. 2, the EBC system provides an EBC broker in each cloud site as well as in the central manager called the multi-site cloud manager. On each cloud site, the EBC broker is included in the cloud portal; each virtual machine runs a daemon named EBC node to communicate with the EBC broker.

EBC node is placed on VM and it receives data through event subscriber and sent data through event publishing. Data is event content in format of key-value pairs. EBC node receives the data if content matching the subscribe expression. There is no address, the subscribe expression can be used to check the identification of EBC node. Other key-value pairs are data container for other information such as commands and arguments. The subscription is actually done through the EBC broker on VM. We implicit the EBC broker on VM make the clear meaning of connection between EBC node and Cloud Portal.

EBC broker: Each site has a portal provide interface to manipulate the system. Portal of the site will be registered to the multi-site cloud manager portal when a site joins system. Each portal keeps a broker act as a router.

Routing between two VMs in different sites: event will be transmitted from VM_A to portal broker and deliver to VM_B via brokers. A matching based on event content-based filter cause data is delivered to the receiver. There is no such a container as in current IP architecture which decides the package delivering by its address, e.g IP address and hostname.

Migrating a running application from VM_A to VM_B includes the following steps:

(1) *Initialization*: VM_A receives a command migrating the application P to VM_B
(2) *Checkpointing*: Images of the application P are taken on VM_A
(3) *Transferring image*: The images are transferred from VM_A to VM_B
(4) *Commitment*: VM_A informs that it has completed transferring images to VM_B
(5) *Restarting*: The application P is recovered and continue its execution on VM_B.

V. IMPLEMENTATION

Virtual infrastructure management in our system is OpenNebula [11]. Such the system includes an OpenNebula front-end and many OpenNebula nodes. The virtual infrastructure management OpenNebula is used to deploy and manage virtual machines. Each virtual machine configuration uses virtual disk image of pre-installed OS and OpenMPI.

A. AppMonC Architecture

AppMonC architecture identifies a system using the event-based checkpointing mechanism. The AppMonC system is located in the gap between higher management level and the virtual infrastructure management, lead to manipulate executing environment on virtual machine. Currently, OpenNebula implements its management functions based on virtual machine hypervisor that limits manipulation at application level. Using AppMonC, we can communicate with MPI programs using OpenMPI library through command line. Besides, the wrapping of AppMonC interface could implement API support higher management level such as scheduling, load balancing, etc.

AppMonC aims to provide high level management functions so that it needs to build an interface give user a sense of connection with system. In this paper, we study the case of OpenMPI program and it may raise a requirement of expanding to other kinds of application. The developed interface could be generic enough to handling later expansion. The notification service need to package not only the event representations but also communication content between user interface and other parts of system.

B. AppMonC Event

The AppMonC system supports two types of event to interact with the outside world and inter-component communication. The first event type is used to manage system as high level event. From outside the system, we could receive system status by sent *sys_info* event. The AppMonC system receives *start_App/stop_App/migrate_App* event and then performs the corresponding actions. The other event type is sent from AppMonC Front-end to AppMonC node called the application driver event. When the system needs to know the status of aliveness of AppMonC, AppMonC Front-end sends out *node_alive_checking* event. *Start/checkpoint/restart/ destroy* events related to OpenMPI checkpoint implementation are defined for the application management purpose.

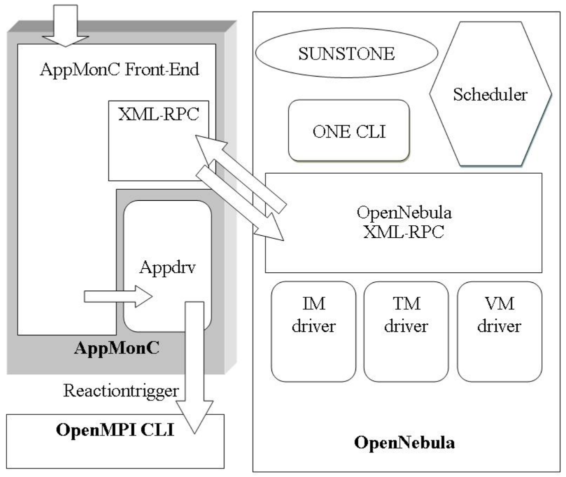

Fig. 3. AppMonC components

C. AppMonC Components

AppMonC separates the component deploy on OpenNebula front-end and virtual machine as in Fig. 3. Both components are implemented as daemons running in background on system.

AppMonC front-end could be seen as event composer can take role of both event consumer and event producer. Being implemented as daemon, AppMonC front-end registers listener port and subscribe system status and application event. Based on analyzing received event, AppMonC front-end prepare and publish event to related AppMonC node.

Appdrv implements the AppMonC node and acts as event consumer. It implements trigger mechanism to invoke MPI program through OpenMPI CLI.

D. AppMonC Administrator

System deployment: includes two phases (i) AppMonC front-end daemon is deployed on OpenNebula front-end (ii) Appdrv daemon is deployed on each virtual machine.

Start/stop system: AppMonC front-end is manually started at Front-end node. Appdrv daemon is automatically started up during boot up using contextualized technique [6] provided by OpenNebula.

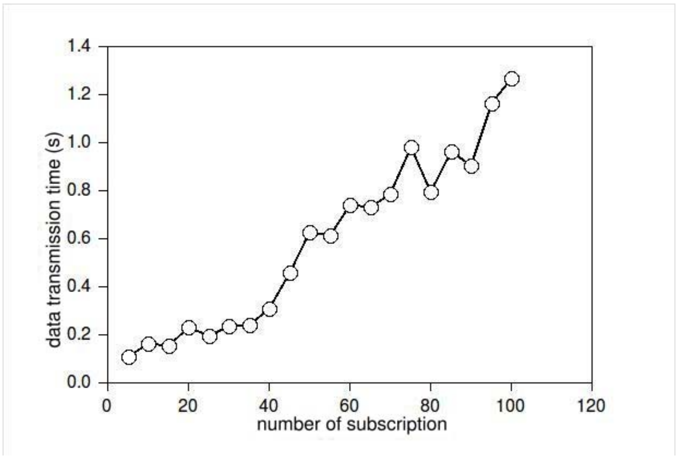

Fig. 4. Event data transmission time

VI. Evaluation

Our experiment is performed on identical hosts with Intel(R) Xenon(R) E5506 processor and 2GB DDR RAM. Each host has a Intel Pro/1000 Gbits/s NIC to transfer data. Storage is accessed via NFS mount system. The guest OS is Linux kernel 2.6.38.

The cloud infrastructure management is OpenNebula version 3.0.0. OpenNebula is configured to use KVM hypervisor. Distributed event-based system using SIENA server version 2.0.1 and C++ client version 0.4.3 with modification to support new mapping table in Siena router 2.x. Migration application is MPI program using library OpenMPI version 1.5.4.

All VMs are configured use 1 processor, 2 GB image size and 768MB of RAM. VM uses the same host's OS which is Linux 2.6.38. Current implement of SIENA library allow transfer string data which promise to implement file transfer mechanism but it need another develop of the library. In our testbed, we implicit the file transferring by set up VM use NFS file system on the MPI folder path.

The experiment of dependency VM migration is much complex and it cannot resolve the dependency among VMs in some cases. We only measure the case of a MPI application run on one VM. As our experiment, the number process of MPI application reach 100 make CPU usage 100% (1 core) so we keep that maximum number of processes.

The evaluation compares the migration time using EBC mechanism and the total VM migration time using OpenNebula migration feature. Due to the problem of time synchronization between two VMs, it makes difficult to measurement the whole process migration which starts in one VM and complete in other VM. So we do the 2-phase measurement includes the data transmission time and the checkpoint/restart time.

Fig. 4 shows the data transmission time related to the increasing number of Pub/Sub subscription. Increasing number of subscription up to 100, the transmission is mostly lower than 1s. Each subscription represents for one DEBs subscriber place on each VM. In comparing with the checkpoint time of 100 processes MPI application around 25s in Fig. 1, the bad case which complex system which has a hundred of subscription transmission time is still less than 5% of checkpoint time.

TABLE I. Migration time

Migration time	
EBC (bad case)	<50s
VM migration using KVM support feature	49-68s

The combination of data transmission time with checkpoint time, we could evaluate the bad case where MPI application has around 100 processes and complex system, the total time still less than 50s which could be consider as the average of migration as in Table I. In the evaluation we include an assumption that restarts time is as same as checkpoint time.

VII. Conclusion

In this article, we introduce the event-based checkpointing tool named EBC supporting application-level migration mechanism on multi-site cloud. The architecture and the implementation of EBC are presented.

The cost of applying EBC is small in comparing with migration the whole VM. Even in the bad case in applying EBC could acceptable and the cost still smaller than VM migration but it produces greater impact on image size reduce and this leads to save tons of bandwidth on multi-site manner. This technology support the following: (i) It enables the communication between two VMs in a different cloud provider; (ii) It ensures that the computing environment of VM can be access in case the cloud infrastructure manager does not know the VM information; and (iii) The cloud infrastructure manager can free up the low payload VM which is useful for other manager tool such as scheduler or power management.

References

[1] A. Carzaniga and A. L. Wolf, "Content-based Networking: A New Communication Infrastructure," Developing an Infrastructure for Mobile and Wireless Systems, pp. 59–68, 2002.

[2] C. Clark et al., "Live Migration of Virtual Machines," in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, vol. 2, USENIX Association, pp. 273–286, 2005.

[3] J. Duell, P. Hargrove, and E. Roman. "The Design and Implementation of Berkeley Lab's Linux Checkpoint/Restart," Technical Report LBNL-54941, Lawrence Berkeley National Lab, 2003.

[4] E.N. Elnozahy, L. Alvisi, Y. Wang, and D.B. Johnson, "A Survey of Rollback-recovery Protocols in Message-passing Systems," presented at ACM Computing. Survey, pp.375-408, 2002.

[5] J. Hursey, J.M. Suyres, T.I. Mattox and A. Lumsdaine, "The Design and Implementation of Checkpoint/Restart Process Fault Tolerance for

OpenMPI," the IEEE International Parallel and Distributed Processing Symposium, pp. 1-8, 2007.

[6] K. Keahey and T. Freeman, "Contextualization: Providing one-click virtual clusters," The IEEE Fourth International Conference on eScience, pp.301-308, 2008.

[7] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky Computing," IEEE Internet Computing, vol. 13, no. 5, pp. 43–51, 2009.

[8] D.S. Milojičić, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou, "Process Migration," ACM Computing Surveys, vol. 32, no. 3, pp. 241–299, 2000.

[9] D. Nguyen and N. Thoai, "EBC: Application-level Migration on Multi-Site Cloud," the 2012 International Conference on Systems and Informatics (ICSAI 2012), Yantai, China, pp. 876-880, 2012.

[10] OpenMPI Homepage. http://www.open-mpi.org/.

[11] OpenNebula: http://opennebula.org/.