EBC: Application-level Migration on Multi-site Cloud

Duy Nguyen and Nam Thoai
Faculty of Computer Science and Engineering
HCMC University of Technology
Ho Chi Minh city, Vietnam

***Abstract*—Cloud computing providers offer their services that help IT organizations to increase hardware utilization dramatically, and to scale up computational infrastructure instantly. The cloud resource management system could suspend/migrate some running jobs/virtual machines in order to reserve resources for other ones to improve resource utilization. In this case, the cost of moving the whole virtual machines is too high due to the big size of virtual machines. Moreover, supporting communication between the two migrated virtual machines is still a hard problem. 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 so it is integrated into cloud infrastructures like OpenNebula easily. EBC supports not only sequential programs but also MPI programs. Moreover, any C&R included in MPI libraries is reused in EBC and so it is independent from MPI library versions. 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

The Cloud computing paradigm is sometimes considered the commercial success of computing resource sharing. Infrastructure-as-a-Service (IaaS) is a type of cloud computing services that provide infrastructure on demand. An IaaS provider typically exposes its provided infrastructure to customers in unit of virtual machine (VM). Consumers could find a suitable IaaS that provides desired (virtual) hardware corresponding built-in service/application. With recent development in cloud computing, it allows to build a system easily and to provide the bigger computational capability at lower costs on clouds.

In this paper, we aim to address the lack of well-structure system support application migrating by proposing EBC - an Event-Based Checkpointing tool - as a conceptual tool for multi-site cloud. Our tool includes addressing mechanism for VM in different sites based on the site portal and the abstract identifier of VM. It leverages software architecture of event-based notification system in linked, identified data to facilitate multi-site cloud. It does not need the understanding of the system; therefore, the system can add or remove a site easily. Finally, we show how to deploy EBC in existing cloud infrastructure manager in order to offer application migration between the two sites. In summary, our paper makes the following major contributions.

- We develop an environment of combining multiple cloud sites using only the site's abstract information.
- We introduce an addressing mechanism to link and identified VM in different sites.
- We propose EBC which is a tool supporting application-level migration on multi-site cloud.

The rest of this paper is organized as follows. Section II presents the review of migration and Internet architecture. In section III we discuss the computing environment on multi-site cloud and some of the challenges for application-level migration. We describe our EBC system architecture in section IV and the migration of application in section V. Section VI evaluates the performance of our technique for MPI application. Section VII concludes the paper and outlines our future work.

II. Migration and the Internet architecture

A. Migration

Process migration is the act of transferring a process between the two machines. Process migration is a hot topic in system research during the 1980s has seen very little use for real-world applications. Milojicic [1] gave a survey of possible reasons include 'residual dependencies' that migrated process retain on the machine from which it is migrated. Application contains one or more process so that the application migration share the same issues with process migration. In other side, migration of the whole virtual machine has been proposed to avoid the problem of residual dependencies [2]. Moving or copying a virtual machine could transfer consistently data in-memory state such as protocol stacks of the OSI 7 layer. Live migration target minimizes the downtime during the migration of virtual machine.

B. Re-architect the Internet

In recent years, there are many researches focus on the issue of host centric design of the current Internet in order to solve the reach ability problem of Internet hierarchical IP address space [3]. A key question is a new thinking "out-of-the TCP/IP-box" for the future network. The reasoning is based on the large scale use of the Internet as connected devices increase rapidly. An IP address, which is used in the network layer as a locator to locate the destination host and forward packets to it, is also used as an identifier in

978-1-4673-0199-2/12/$31.00 ©2012 IEEE

the transport and application layers to identify the host or communication sessions [4]. The explicit separate the owner ship of the hosts from the underlying connectivity provides capability to express organization policies separately from the underlying infrastructure routing policies [5].

Information-Centric Networking (ICN) is one of the significant direction of current networking research. The principal ICN paradigm is not end-to-end communication among hosts as in the current Internet architecture [6]. ICN considers pieces of information as network entities rather than only indirectly identifying and manipulating them via the node hosting that information [7]. ICN shares the concept with information/content/data-oriented/centric network, the flow of message is driven by the node that expresses their interest and the information identifier of the message. In this way, information become independent from device container [3].

The publish/subscribe paradigm [8] is a promising trend to provide a communication for information-centric system. Publish/subscribe system has been widely studied in manner spatial and temporal decoupling. The benefit of moving publish/subscribe downward network stack is one of the challenging objectives of interest-driven architectures [3]. On-demand content delivery service is important form of communication in content-based network [9]. Despite their different, both forms of communication would benefit from an addressing scheme based on content rather than on more or less flat identifiers which mean that both should be integrated to some extent within a unified content-based routing infrastructure [6]. By ability of share underlying platform, publish/subscribe event notification can be directly mapped to common content-based network layer and natively support feature of information-centric network. It enables the capability of applying the principle paradigm on a well developed publish/subscribe system like Distributed Event-based system (DEBs) [10] [11].

III. Application-level migration motivation

A. The separation of multi-site cloud environment

A combination of various cloud providers keep us from making an investment in computing hardware while ensure system utility. Multi-site cloud supports users for migrating their workload to different cloud vendors, preventing vendor lock-in [12]. The multi-site cloud system is created by joining of various public/private clouds.

Since clouds are physically separated, the most basic requirement for multi-site cloud is to handle long-distance migration among hosts serrated across the WAN. But the cloud is also logically separated in distinct administrative domain. Otherwise, a cloud site does not assist its users to use resources from other sites. It is because of commercial interest, a cloud provider usually has a proprietary system where VM created from image and run only on provider's site [13]. If cloud paradigm is to be extended to use multiple providers, several challenges need to be resolved. One of the requirement multi-site cloud discussed in this paper is the capability to create efficiently migrating mechanism.

B. The need of application-level migration

Although it resolves some problems of process migration, migration of the whole virtual machine has some limitations. It costs a lot of time and consumes network bandwidth due to the big size of virtual machine. While process migration only migrates the image of the process on memory and so the migration time and the downtime is considerably shorter [14]. Virtual machine migration does not ensure to keep the consistency of applications running on multiple virtual machines. Instead, application embedded the checkpoint capability can support the consistent state amongs its processes running on different virtual machine.

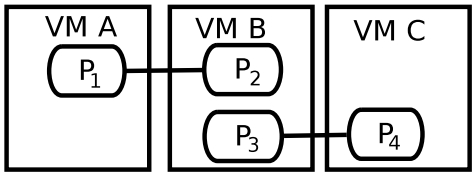

Fig. 1: Migration dependency of VMs

There are some cases in which virtual machine cannot be applied. For example, an MPI application consists many processes running on different VMs; because of the communication between the two processes, MPI application is usually deployed on VMs placed in the same site. Figure 1 shows the case of dependency between the two applications is difficult to migrate. To migrate application (P_1, P_2) it needs to migrate both the VMs A and B. When B is migrated, the dependency between the process P_3 and the process P_4 require the migration of C. Using application-level mechanism, it only needs to migrate the application (P_1, P_2).

The other case of non-applicable virtual machine migration is the reducing of number of running VMs. Due to the narrow dynamic power range of servers, even completely idle servers still consume about 70% of their peak power [15]. It makes sense a management tool migrate application from some low payload VMs to other VMs. Full payload VMs could help release some low payload VMs by relocating their applications.

A simple implementation of process migration is checkpointing processes and then transfering images to target in order to recover processes. There are many implementation of MPI supporting checkpoint capability [16] [17] [18]. The embedded checkpoint mechanism helps to overcome issues of process migration. To migrate an application, the transmission subsystem needs to know address of source and destination. In multi-site cloud, a VM is created from scratch and change dynamically that cause the requirement of addressing VM.

C. Challenges of application-level migration

Virtual machine monitor (VMM) is the module in abstract cloud architecture provide framework allow virtual machine to run [19]. VMM relies on a library called libvirt which provide facility to start and stop VMs. The VM definition is

passed to VMM when it created. The information of VM, e.g. CPU, RAM, can be collected but not the computing environment on VM. When a VM is started, its state change to RUNNING status, which actually embed various status of execution environment inside VM to be set up. It may start up networking interface or complete the boot up progress. Eventually, we know the MAC address of network interface but not the IP address.

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 current cloud network is based on IP protocol. IP address includes both locator and identification. The IP address is global unique make some limitations. It is not suitable to deploy on multi-site cloud with requirement of scalability and mobility. In multi-site cloud, each cloud manage VM using VMID (identification of VM) but it is lack of the IP address because of VMM limitation. The addressing problem exists not only in side 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.

One solution for the network in multi-site cloud is proposed in sky computing [20]. The user-level routing mechan-ism, called ViNe, transfers packets between different cloud sites. ViNe router provide all-to-all communication but it still relies on IP address. Other solution based on multicast but it requires information of the network structure. Selection capability offered by multicast addresses is limited. Instead, target of packet can be everything not only one team.

IV. EBC: The Event-Based Checkpointing

In this section, we will introduce the architecture of EBC, our proposed application-level migration tool for multi-site cloud. EBC tool includes all-to-all communication toward address-agnostic and node initiator connection. It provides mechanism to communicate between the two VMs even though their information have not been collected by portal or multi-site cloud manager. This communication subsystem, bases on publish/subscribe event notification, helps overcome the limitation of VMM module in cloud infrastructure manager. The connection direction, start from EBC node to EBC broker, is applicable to the context of VM in cloud infrastructure. Moreover the connection from the slave broker to master broker share the same model of joining cloud to multi-site cloud system. EBC tool includes separated entities: nodes and brokers.

A. EBC broker and routing construction

Each site has its own portal where an EBC broker is deployed. When a site join system, its portal is registered to the multi-site cloud manager portal. The progress of EBC broker interacting follows these steps:

1) An EBC broker is started on multi-site cloud manager.
2) An EBC broker is started on each cloud portal. VMs subscribe to its cloud portal's EBC broker.
3) As a site joins the system, the cloud portal's EBC broker reconnect to the multi-site cloud manager's EBC broker.

The event routing between two different sites: events are transmited to source portal's EBC broker and then deliver to destination portal's EBC broker after routing through Distributed Event-based Systems (DEBs) network. There is no such thing as a container e.g. IP address, hostname; and the routing is built upon event data content. This routing adapt with dynamic computing infrastructure by changing the filter rule.

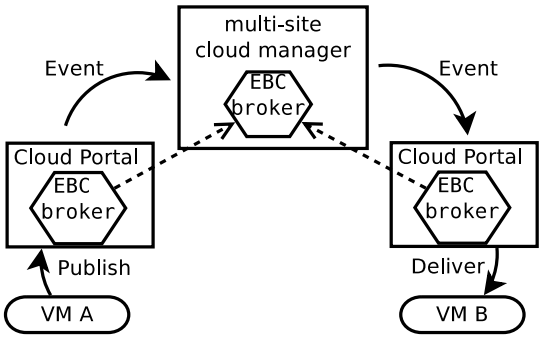

Fig. 2: EBC broker

B. EBC node

EBC node is placed on VM and it receives data through event subscriber and sends data through event publishing. Data is event content in format of key-value pairs. Some of these pairs are used as content matching subscribe expression. The matching between the event content and the filter rules make data delivered. The subscribe expression can be used to check the identify of EBC node. Other key-value pairs is data container for other information such as command and argument. The subscription actually implements through EBC broker on VM. For the simplicy of the figure, we implicit the EBC broker on VM make the clear meaning of connection between EBC node and Cloud Portal.

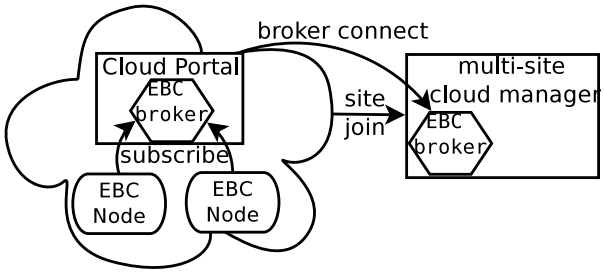

Fig. 3: EBC node

V. Application migration in EBC

This section describes the design of our application migration approach. Our method does not use the IP address which embeds the both kinds of address. The address in EBC system is isolated to locator and identifier. It can reuse the infrastructure information to locate and identify the VM.

The data in EBC are content-based so that the delivering of event to subscriber depend only on its content. Using matching subscribe expression of *addr_key*, EBC node can check the data address match the VMID or site address (cloud indentifier). VMID matching ensures data is sent only to target VM. The site address matching in case data broadcast to the whole site. The *addr_key* could be expanded to support more kind of logical group of VMs.

Figure 4 shows the whole process migrating an application from VM A to VM B. The migration process as a transactional interaction between two VM involved the following phases

1) **Initialization**: VM A receives a command to migrate the application P to VM B.
2) **Checkpoint**: VM A checkpoint the application P.
3) **Transfer application image**: images are tranfered from VM A to VM B.
4) **Commitment**: VM A informs that it has completed transfering images to VM B.
5) **Restart**: the application P is restarted and continue its execution on VM B.

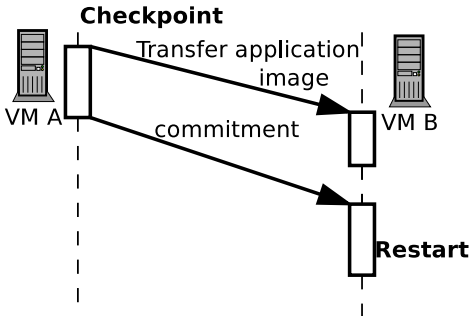

Fig. 4: Process of application migration

VI. Evaluation

A. Testbed setup

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 using KVM hypervisor. Distributed event-based system using SIENA server version 2.0.1 and modified C++ client version 0.4.3 supporting the new mapping table of Siena router 2.x. Applications are MPI program using library OpenMPI version 1.5.4.

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

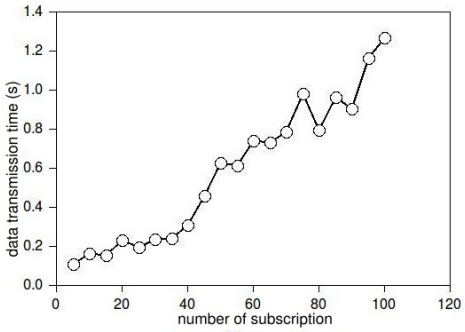

Fig. 5: Event data transmission time

B. Migration time

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 including the data transmission time and the checkpoint/restart time.

Figure 5 shows the data transmission time related to the increasing number of publish/subscribe subscription. Increasing number of subscriptions upto 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 Figure 6. Subscription transmission time is still less than 5% of checkpoint time.

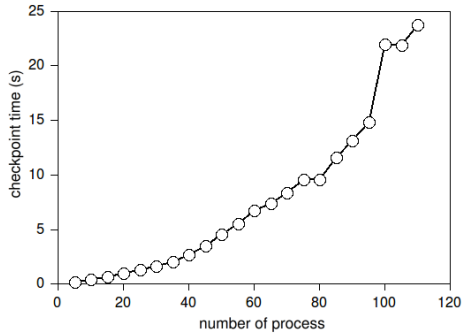

Fig. 6: Checkpoint time

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 considered 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.

TABLE I: Migration Time

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

C. Checkpoint image size

The evaluation compares the application performance of the size of checkpoint image. We compare the MPI application checkpoint image size with the VM checkpoint image size using OpenNebula migration feature. OpenNebula command will call the save/restore command provided by KVM hypervisor. We did not compare case by case between the MPI application checkpoint with VM checkpoint in various number of MPI processes because the VM checkpoint image size is approximate 2.3 GB. Instead, we measure the size of application checkpoint image and its memory size in RAM.

Figure 7 shows MPI application checkpoint image is smaller than MPI application size in RAM. The biggest size of MPI application is only approximate 50% of the whole VM image size, the MPI checkpoint image is much smaller than VM checkpoint image.

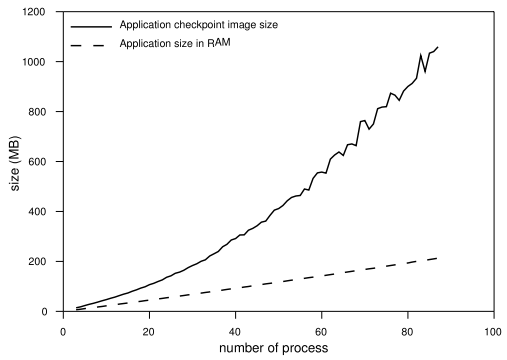

Fig. 7: Comparision of MPI application checkpoint image size and MPI application size in RAM

VII. Conclusion

In this article, we present our application-level migration mechanism which is able to deploy on multi-site cloud. The cost of applying EBC is small in comparing with migration the whole VM. Even in the bad case, applying EBC could acceptable and the cost still smaller than VM migration but it produces a greater impact on image size reducing and this leads to save tons of bandwidth on multi-site manner. This technology supports the followings: (1) It enables the communication between two VMs in different cloud providers; (2) It ensures that the computing environment of VM can be accessed in case the cloud infrastructure manager does not know the VM information; and (3) The cloud infrastructure manager can free up the low payload VM which is useful for other manager tool such as scheduler or power management.

We are currently studying to extend our communication platform to support multiple connections between the two network entities. This innovation enables fault tolerant of network communication. To leverage ICN feature, our approach use DEBs where the connection is initialized by producer. We also investigate how to apply our virtual network service and migration with the on-demand data delivery model of content-based network where the data transmission is initialized by consumer. It will draw more complete picture of ICN behavior on multi-site cloud.

References

[1] D. Milojičić, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou, "Process migration," *Computing*, vol. 32, no. 3, pp. 241–299, 2001.

[2] C. Clark *et al.*, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, no. Vmm. USENIX Association, 2005, pp. 273–286.

[3] C. Esteve, F. Verdi, and M. Magalhaes, "Towards a new generation of information-oriented internetworking architectures," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, p. 65.

[4] V. Kafle, H. Otsuki, and M. Inoue, "An ID/locator split architecture of future networks," in *Innovations for Digital Inclusions, 2009. K-IDI 2009. ITU-T Kaleidoscope:*. IEEE, 2009, pp. 1–8.

[5] S. Paul, J. Pan, and R. Jain, "A future internet architecture based on de-conflated identities," in *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*. IEEE, 2010, pp. 1–6.

[6] D. Kutscher *et al.*, "10492 Abstracts Collection – Information-Centric Networking," in *Information-Centric Networking*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2011.

[7] M. Söllner and B. L. Alcatel-lucent, "Content, Connectivity, and Cloud: Ingredients for the Network of the Future," *IEEE Communications Magazine*, pp. 62–70, 2011.

[8] P. T. Eugster, P. a. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, Jun. 2003.

[9] T. Zahariadis, "Architectures for Future Media Internet," *on User Centric Media*, 2010.

[10] G. Mühl, L. Fiege, and P. Pietzuch, *Distributed event-based systems*. Springer, 2006.

[11] A. M. Hinze and A. Buchmann, *Principles and Applications of Distributed Event-based Systems*. Information Science Reference, 2010.

[12] M. Armbrust, R. Katz *et al.*, "Above the Clouds : A Berkeley View of Cloud Computing," *Science*, vol. 28, 2009.

[13] M. Schmidt *et al.*, "Efficient distribution of virtual machines for cloud computing," in *18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*. IEEE, 2010, pp. 567–574.

[14] T. Maoz, A. Barak, and L. Amar, "Combining Virtual Machine migration with process migration for HPC on multi-clusters and Grids," *2008 IEEE International Conference on Cluster Computing*, pp. 89–98, 2008.

[15] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 13–23.

[16] Q. Gao, D. K. Panda *et al.*, "Application-Transparent Checkpoint/Restart for MPI Programs over InfiniBand," in *Proceedings of the 2006 International Conference on Parallel Processing*. IEEE, 2006, pp. 471–478.

[17] C. Coti, F. Cappello *et al.*, "Blocking vs. non-blocking coordinated checkpointing for large-scale fault tolerant MPI," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*. ACM, 2006, p. 127.

[18] J. Hursey *et al.*, "The design and implementation of checkpoint/restart process fault tolerance for Open MPI," in *IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2007, pp. 1–8.

[19] P. Sempolinski and D. Thain, "A comparison and critique of Eucalyptus, OpenNebula and Nimbus," in *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2010, pp. 417–426.

[20] K. Keahey, J. Fortes *et al.*, "Sky Computing," *IEEE Internet Computing*, vol. 13, no. October, pp. 43–51, 2009.