I.J. Intelligent Systems and Applications, 2011, 1, 58-66
Published Online February 2011 in MECS (http://www.mecs-press.org/)

Sensitive Data Protection Based on Intrusion Tolerance in Cloud Computing

Jingyu Wang[1,2] and xuefeng Zheng
[1]School of Information Engineering, University of Science and Technology Beijing, Beijing, China
Email: btu_wjy@imust.cn,zxfxue@263.net

Dengliang Luo
[2]Information and Network Center, Inner Mongolia University of Science and Technology, Baotou, China
Email: luoimust@gmail.com

***Abstract*—Service integration and supply on-demand coming from cloud computing can significantly improve the utilization of computing resources and reduce power consumption of per service, and effectively avoid the error of computing resources. However, cloud computing is still facing the problem of intrusion tolerance of the cloud computing platform and sensitive data of new enterprise data center. In order to address the problem of intrusion tolerance of cloud computing platform and sensitive data in new enterprise data center, this paper constructs a virtualization intrusion tolerance system based on cloud computing by researching on the existing virtualization technology, and then presents a method of intrusion tolerance to protect sensitive data in cloud data center based on virtual adversary structure by utilizing secret sharing. This system adopts the method of hybrid fault model, active and passive replicas, state update and transfer, proactive recovery and diversity, and initially implements to tolerate F faulty replicas in N=2F+1 replicas and ensure that only F+1 active replicas to execute during the intrusion-free stage. The remaining replicas are all put into passive mode, which significantly reduces the resource consuming in cloud platform. At last we prove the reconstruction and confidentiality property of sensitive data by utilizing secret sharing.**

***Index Terms*—Cloud Computing, Virtualization, Intrusion Tolerance, Cloud Security, Virtual Adversary Structure**

I. INTRODUCTION

Cloud computing distributes computing tasks to virtual resource pools which are constituted of a large number of computers, and cloud computing ensures that various applications can get access to computing power, storage space and various software services when needed. Service integration and supply on-demand coming from cloud computing can significantly improve the utilization of computing resources, reduce power consumption of per service, and effectively avoid the error of computing resources. Cloud computing has the following advantages: super-massive scale, virtualization, high reliability, versatility, high scalability, on-demand service and very low-cost [1]. Notwithstanding there are so many advantages, cloud computing is still facing the following challenges.

A. Intrusion tolerance of the cloud computing platform

In cloud computing mode, all the business process will be completed on the server side. So cloud computing platform will become the attacking core of intruders. In addition, cloud computing platform itself also faces a range of unexpected errors. For example, in the first half of 2010, the cloud computing service of Amazon arouse malfunction due to human error and unexpected power failure, which resulted in a small number of users in the eastern United States losing services and led to a very small number of data loss. So if the server once has problems, it will cause all the services not to run normally and the data not to be accessed.

B. Intrusion tolerance of the sensitive data of new enterprise data center

The most two important issues of cloud computing are confidentiality and security. Enterprises store sensitive business data, such as corporate information and customer information, etc., into platforms of cloud service providers to reduce economic-cost by giving up the control of some data. Therefore, enterprises are bound to worry about the security of data in cloud whether it will leak and whether it will be stolen or not.

For the issues of confidentiality, integrity, availability and reliability of cloud computing, traditional security technologies, such as intrusion detection, firewall, encryption and decryption, etc., can not completely guarantee them, but a survivable technology that is intrusion tolerance technology can make up well. This paper will address the intrusion tolerance problem of cloud computing platform and new enterprise data center by researching on the related intrusion tolerance technology, such as virtualization replicas group and secret sharing.

II. RELATED WORKS AND OUR SOLUTIONS

The main idea of intrusion tolerance is utilizing hardware or software fault tolerance technology of

Nature science fund project of Inner Mongolia (No: 20080404ms0903) and spring sunlight project of ministry of education (No: Z2009-1-01044).

distributed system to mask the impact of any intrusion or attack on the system function, which guarantees the security and continuity of the core function of system. Today's institutions and organizations research on intrusion tolerance mainly adopt technologies of threshold cryptography, secret sharing and distributed redundancy replication, etc., on the real hardware platform. All those technologies require systems with high computing power and high storage capacity, and these resources generally can not be partitioned and they are usually designed for specific applications. Therefore the utilization of resources is low, the management is hard to implement, and the versatility is poor. In addition, the system also has higher requirements for the number and quality of redundant components, and the recovery of redundant components will seriously affect the availability of system service. Therefore, this high cost of intrusion tolerance is unacceptable for cloud computing service providers and users, and this is not suitable for the virtualization technology of cloud computing.

Virtualization technology has been fully developed since it appeared in the 1960s, but its application is still mostly about the efficient management of resources. In fact, virtualization technology also can be used to build security systems, such as Hans P.Reiser and Rüdiger Kapitza, etc., [2] had researched on the intrusion tolerance system with virtualization technology. With the proposal of cloud computing, researching on the security of cloud computing based on virtualization technology has become critical.

In order to address the security, reliability and availability of cloud computing platform, this paper presents a solution of virtualization intrusion tolerance based on cloud computing (CC-VIT). This method allows the system to tolerating F faulty replicas in N=2F+1 replicas and ensure that only F+1 active replicas to execute during the intrusion-free stage. The remaining replicas are all put into passive mode, which significantly reduces the resource consuming in cloud platform. While the traditional Byzantine fault-tolerant algorithm tolerates F faulty replicas require N = 3F +1 replicas.

For the sensitive service data of data center in cloud computing can be protected by secret sharing. The current proposed secret sharing schemes are mostly based on access structure [3, 4]. As access structure often describes the authorized subset of system, while the system configuration (properties) and security requirements describe the system threats, and its own vulnerabilities and the security target should be achieved in this system configuration, etc.., so it is very difficult to determine the corresponding access structure. While adversary structure [5, 6] indicates the set of participants which is permitted to be simultaneously compromised in system configuration, which can be directly determined by the system configuration (properties) and its security requirements. For these reasons, we propose the concept of virtual adversary structure based on CC-VIT model, and then present a method of intrusion tolerance to protect sensitive data in data center of cloud computing based on virtual adversary structure. This method partitions sensitive data into segments (partitions) and distributes them to each participant, and ensures that only legitimate participants can associate with their partitions to reconstruct sensitive data. At the same time, the illegal participants associate with their partitions can not obtain any information about sensitive data. All participants in this proposal are virtual machines which run on the virtual machine monitor, so it saves a lager number of physical servers. So this method is able to guarantee the security and integrity of important data or sensitive data, and also can effectively reduce the consumption of physical resources.

III. CC-VIT Prototype

A. CC-VIT System Structure

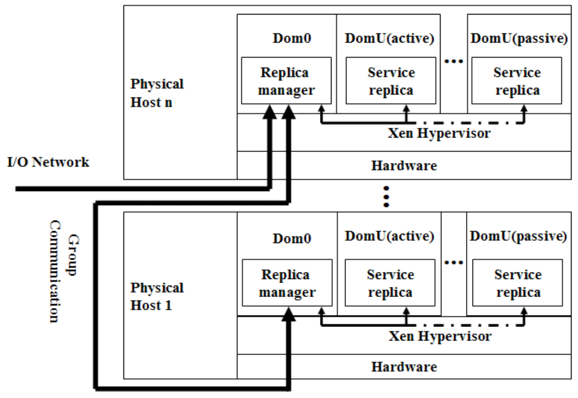

Figure 1. CC-VIT system structure

This paper adopts virtualization technology to realize the intrusion tolerance of cloud platform, and its system prototype is shown in Fig. 1. Hypervisor is a very small software layer which is running on the top of hardware and that is also virtual machine monitor. Over the Hypervisor, we divide different domains. In fact these are many virtual machines running on Hypervisor, while service instances are running in the customer domain DomU(Guest Domains)，Guest Domains have passive mode and active mode. Privilege domain, which is Dom0, controls the creation, execution and destruction of client domains.

Replica manager consists of the following components:

- Basic component that is used to accept client requests;
- Group communication system that is used to continuously forward requests to all the replicas;
- Replication logic that is used to realize proactive recovery function;
- Voter component that is used to activate passive replicas on-demand;
- Related mechanism that is used to update states;
- Related component that is used to clone replica domains.

CC-VIT provides communication through Hypervisor. Replication logic is running in privilege domain Dom0, which is in charge of specific tasks of service instances while specific service replicas are running in the isolated guest domains. The interaction of clients and servers is realized by the replica manager which is running in system domain Dom0.

The reasonably design of CC-VIT is based on the following assumptions:

- The interaction of clients and remote services is based on the basis of request/reply network messages, and it is intercepted at the network level.
- The remote service can be modeled as a deterministic state machine.
- Service replicas, including operating system and execution environment, may fail in Byzantine failure; CC-VIT permits at most $f < \left\lfloor \frac{n-1}{2} \right\rfloor$ replicas fail within a single recovery period.
- The other system parts, including Hypervisor and trusted system domain, fail only because of the crash of physical host.

In this paper, we call the set of virtual machines which permit to be simultaneously compromised in system configuration as virtual adversary structure. Virtual adversary structure indicates the subset of participants which is permitted to be simultaneously compromised in intrusion tolerance system, which is also the number of virtual machines (Guest OS or DomU) in CC-VIT model.

B. *Work Process of System Service*

In the CC-VIT there is a physical machine that is used to accept client's requests, and the replica manager of this machine intercepts client's connection requests and forwards these requests to all the replicas of replica group through group communication system. Each replica executes client's requests, and then returns executive results to the physical machine node which accepts client's connection requests.

When the physical machine has received a large number of returned results, then replica manager, which runs in system domain Dom0, executes majority voting algorithm to determine a correct result, and returns the results to users. The realization of the whole process is transparent to users. Replication logic specifically realizes the instantiation and initialization of service replica.

IV. Intrusion Tolerance of The Cloud Computing Platform

A. *Hybrid Fault Model*

The isolated execution environment permits implementing heterogeneous fault and intrusion tolerance model. That is to say, this mode can tolerate malicious attacks on the application instances in the application domains, such as the Byzantine failure, and it can tolerate the crash of physical host in system domain and Hypervisor. In this hybrid fault model, service replicas which run in the isolated application domains, including operating systems, middleware structure and implementation services, can all be heterogeneous.

Redundant Execution on a Single Host (RESH) [7] is a variant application of CC-VIT model. It permits redundantly executing multiple replicas of service on a single physical host. This redundant execution needs virtual machine monitor to create multi-application domains to realize isolated execution of replica instances. RESH permits tolerating non-benign random faults in replicas, such as undetected bit errors in memory, and as well as to tolerate software faults by using N-version programming.

Redundant Execution on Multiple Hosts (REMH) is another variant application of CC-VIT model. It permits replicas with same core architecture to redundantly execute on multiple physical hosts. The main difference compared with RESH is that REMH integrates a group communication facility in the replication logic. Group communication module is responsible for the communication and coordination between replicas in a group of physical hosts. REMH allows tolerating full crashes of some of the physical hosts running replicas.

B. *Active Replicas and Passive Replicas*

Traditional Byzantine Fault Tolerant (BFT) replication commonly needs executing client's requests on every replica. In order to reduce resource consumption, we introduce the concept of passive replicas to CC-VIT model. During the normal execution of workloads, we need at least F+1 active replica to detect errors and the remaining replicas are all put into passive mode. If an error (accident) occurs in active replica or return timeout, another passive replica will be activated to take over. To make this method feasible, the system architecture must allow rapidly activating passive replica. Otherwise, the services will not be available for customers.

In this system, virtual machines hosting passive replicas are put into suspended mode. They are only allocated with memory and disk, but not allocated with other resources, such as CPU or bandwidth. When activating the passive replicas, replica manager simply activates the corresponding virtual machine, and this process requires only a few hundred milliseconds. In general, to tolerate F faulty replicas, this system only requires 2F +1 replicas. But in order to tolerate an unlimited number of replica faults during the whole system life-cycle, we should adopt proactive recovery strategy to continually resume faulty replicas when faulty replicas is about to break through threshold F. Therefore, as a matter of fact the system needs 2F +2 replicas, and the extra one replica is passive replica, which will be used to clone replica domain and realize proactive recovery.

C. *State Updates and State Transfer*

In a stateless replication system, the transfer from old replica to new one is almost instantaneous. While in a state-full system, after creating a new instance of operating system, middleware, and service, the new replica needs to be initialized with the service state, which requires the support of state transfer. State transfer increases the unavailable time of service replica in the process of proactive recovery, as request execution has to be transitorily stopped during the creation of state checkpoint. Furthermore, the state transfer also affects the utilization of network and CPU resources.

Virtualization technology can realize storage management of disk through virtual machine monitor. In the virtualization disk management, permanent state storage of virtualization can be used to realize effective state re-mapping mechanism among multiple domains. This mechanism can be used to realize state transfer strategy with low cost and low overhead. State transfer must deal with the heterogeneity between different replicas, while heterogeneity is usually introduced by diversity. Transferring the state into local replica-specific representations needs some support of replica implementations [8].

In CC-VIT system, in order to process the suspended requests, passive replica also must have the recently available application state after activating passive replica. In order to address the problem of state updates and state transfer, this system adopts regular state updates method to update passive replicas. In general, after executing a modifying request, replica managers will retrieve state updates, and the state changes are triggered by the requests from all active replicas. After that, replica managers agree on these state updates and store the verified version of each state updates in a local log. In order to guarantee the correctness of verified versions, we can create distributed checkpoints, and adopt the method of checking local state checksums to verify the validity of local state. Only in the case of an invalid local state, a more expensive remote state transfer is necessary. When the list of log size reaches a certain limit, replica managers temporarily activate local passive replicas to execute the accumulated state updates. After completing state updates, replica managers re-suspend passive replicas which have just been updated, and then replica managers truncate their logs. Note that these periodic updates of passive replicas can reduce the load of instantaneously updating a large number of states on the occurrence of faults, because when a passive replica is activated to tolerate a fault, only state updates after the last periodic update need to be executed.

D. Proactive Recovery

Traditional intrusion-tolerant systems [9, 10] generally adopt Byzantine fault tolerant replication algorithms to tolerate a finite number of F replica faults in group of N replicas. But this method has a potential problem. As long as be given sufficient time, an adversary might be able to compromise more than F replicas and break through the threshold F that system can tolerate, which ultimately makes intrusion tolerance system fail. The method to solve this problem is periodicity reinitializing the faulty replicas from a secure base, and the core idea is proactive recovery strategy. Proactive recovery approach can remove some potential intrusions. If we adopt proactive recovery strategy, as long as the number of replica failures that the system can tolerate is limited in the threshold F within a single recovery period, the system can tolerate unlimited replica failures over the system lifetime.

In the traditional proactive recovery strategy, we only permit some parts of replicas to restart at the same time to avoid the unavailability of services, while adopting virtualization technology can realize all the replicas simultaneously resume but hardly affect the execution of services. As any virtualization technology provides a core function for creating and destroying domains, so we can achieve an efficient proactive recovery strategy through this mechanism [8]. Periodical recovery strategy can be triggered by the resume service in the isolated system domain. In order to reduce the unavailable time of system services, we can adopt a parallel recovery strategy that creates a new domain in parallel to the execution of the other applications.

In the method of virtualization technology, replication logic runs in an isolated and intrusion-free system domain. So replication logic can be used as a trusted entity that is able to completely re-initialize the replica domains without requiring the support of any dedicated hardware. The recovery strategy initializes the complete replica to a “clean” state, securely obtains the service state from other replicas through a fault-tolerant state transfer protocol, and then transfers to the corresponding service state to execute tasks allocated by system. Hypervisor is able to shut down and restart the replicas running in virtual machines. When in the execution of proactive recovery, the new replica instance can be started by using an additional virtual machine in parallel to the execution of the old replica. This method can substantially reduce the downtime of service during recovery, as the boot process does not affect the availability of replica. After initialing the new replica, the replica manager can shut down the old replica and trigger the activation of the new one [11].

However, virtualization-based proactive recovery also has the following shortcomings. On the one hand, proactive recovery usually has some negative impact on services, as the limited local resources (such as CPU and memory) have to be shared among all the replica domains. Proactive recovery inevitably involves the creation of new domains and the destruction of old domains, which inevitability consumes the limited resources. On the other hand, this proactive recovery strategy significantly reduces the unavailable time of service when replica transfers state from an old one to a new one.

In fact, in this system, the passive replica has the latest available application state. So the passive replica can be used as the basis of proactive recovery. When the faulty replicas will break through the threshold F, proactive recovery is triggered. In this system, we could clone a passive replica to quickly create a new replica domain with the latest available state, and then obtain the state after the last state updates through state transfer. After state updates and state transfer, the replica manager hands over request execution to the former passive replica (new replica) and shuts down the old faulty active replica. In this way, a potentially faulty replica is efficiently replaced with a clean one, and this system also effectively reduces the busty negative effects of proactive recovery on the system platform.

E. Diversity

Virtualization technology simplifies the introduction of system diversity, as there runs replication logic in the system domain which fully controls the operating systems and the instances of services. The diversity of replicas is the essential technology in intrusion tolerant systems, as it can avoid an adversary to exploit the same vulnerability multiple times within a short time interval. However, virtualization-based recovery mechanism can easily introduce the basis ideal of diversity both in space and in time [12].

The hypervisor-based replication architecture allows transparently intercepting the interaction of client–service, independent of the guest operating system, middleware, and service implementation. As long as the assumption of deterministic service state is not violated, the service replicas may be completely heterogeneous, which include different operating systems, middleware, and service implementations. Virtualization -based recovery can be used to provide with diversity in time. For example, the operating system can be changed in the new version, or internal configurations can be modified on each recovery reboot. Address space randomization is another popular technique which is introduced to intrusion tolerance system through diversity.

V. Intrusion Tolerance of The Sensitive Data of New Enterprise Data Center

Secret sharing [13, 14] partitions sensitive data or critical data into multiple security segments (partitions) and distributes them to the collaborating multiple distributed servers set, and ensure the security and integrity of secret or critical data even if parts of system is compromised. Therefore, realizes the decentralization of risk and intrusion tolerance. The participants of secret sharing share a secret S in set P. If we distribute each participant with a segment (partition or shadow) of secret S, we ensure that only legitimate participants can associate with their partitions to reconstruct secret S (the property of reconstruct secret). At the same time, the illegal participants associated with their partitions can not obtain any information about sensitive data (the property of perfect confidentiality), and then we claim such a secret sharing scheme as perfect.

Let P={$p_1, \cdots, p_n$} be the finite set of all virtual machine identifiers (participants) in the set of virtual machines, and then the virtual adversary structure β is a family of subsets of P that specifies which participants the adversary may corrupt at the same time. It can be informally represented as β = {X|X$\subset$P; X is the subset of participants which permit to be simultaneously compromised}. β is monotone, if S$\in\beta$ and T$\subset$S then can export T$\in\beta$, and can be uniquely determined by the corresponding maximal virtual adversary structure $\beta_{\max}$. The maximal virtual adversary structure $\beta_{\max}$ means that no subset in $\beta_{\max}$ contains another one, namely $\beta_{\max}$ = {X|X $\subset$P; X is the subset of participants which permit to be simultaneously compromised; $\forall$X$_1$, X$_2$ $\in \beta_{\max}$, X$_1 \not\subset$ X$_2$ and X$_2 \not\subset$ X$_1$}.

Definition 1: 2^P is the power set of P, then $\beta \subseteq 2^P$ is a virtual adversary structure, if β satisfies X $\in \beta$, X'$\subseteq$X $\subseteq$P $\Rightarrow$ X'$\in \beta$.

Definition 2: Given the virtual adversary structure β over participants set P, the corresponding maximal virtual adversary structure is $\beta_{\max}$ ={B $\in \beta$ |B$\not\subset$B',$\forall$B'$\in\beta$ }.

To expediently denote virtual adversary structure, we introduce threshold gate Θ_k^n.

Definition 3: Set P={$p_1, \cdots, p_n$} as the input of Θ_k^n, then $\Theta_k^n = 1 \Leftrightarrow \exists$ P $_{i1}$, P $_{i2}$,…P $_{ik}\in$P, and P $_{i1}$ =P $_{i2}$=…=P $_{ik}$=1. Namely the output of the gate is 1 only when n input at least k is 1. Obviously, AND gate and OR gate be equivalent to Θ_n^n and Θ_1^n, while the traditional threshold scheme can be denoted as g(S) =Θ_{t+1}^n(S).

Definition 4: Let P={$p_1, \cdots, p_n$} be the finite set of all participants, K is system secret, then a secret sharing scheme about K can be denoted by a mapping Π: K$\rightarrow$ g(p_1) $\times \cdots \times$ g(p_n), among which g(p_i) is the shares collected by the participant p_i ($p_i \in$P). We say that Π is the scheme realizes adversary structure β if it satisfies the following two properties:

(1)Reconstruction Property.$\forall$X $\subseteq$P, if X$\notin\beta$, then H(K|**{**g(p_i| $p_i \in$X)**}**=0, where H(•) denotes entropy.

(2) Perfect Property.$\forall$X$\in\beta$, H(K|**{**g(p_i| $p_i \in$X)**}**= H(K'|**{**g(p_i| $p_i \in$X)**}**, where K' denotes any element in secret space.

We define the following rules for facilitation, □X, X□P, X can be represented by an n-tuple, where n=|P|. That is X= (x_1 , x_2 …x_n) $\in$ {0, 1}^{n}, where $x_i = \begin{cases} 1, & P_i \in X \\ 0, & P_i \notin X \end{cases}$; $\overline{X}$ = ($\overline{x_1}$, $\overline{x_2}$ … $\overline{x_n}$) $\in$ {0,1}^{n}, where $\overline{x_i} = \begin{cases} 1, & P_i \notin X \\ 0, & P_i \in X \end{cases}$.

The secret sharing based on virtual adversary structure β is as follows:

Given the system secret K, the allocation of shares can be described in the following steps:

Step 1: Compute $\beta_{\max}$ = {B $\in \beta$ |B $\not\subset$ B', $\forall$ B' $\in \beta$ }.Without loss of generality, marking $\beta_{\max}$ = {$B_1, \cdots, B_m$}, where m=|$\beta_{\max}$|;

Step 2: Secret K is split into $K_1, \cdots K_m$, and satisfying

$$K = \sum_{i=1}^{m} k_i \cdot$$

Step 3: Let ω_β = {W_1, $W_2, \cdots W_n$} , where $W_i = \overline{B_i}$, 1≤i≤m. We call ω_β as the write structure of virtual adversary structure β .

Step 4: Distributing K_i to each $W_i \in \omega_\beta$. In this way, $\forall p \in P$, the shares belong to p are $g(p)=\{K_i | p \in W_i, 1 \leq i \leq m, p \in P\}$.

VI. Results and Experiment Analysis of CC-VIT System

In order to verify the feasibility of CC-VIT system, we realize the VM-FIT prototype on Dawning server (Intel(R) Xeon(TM) CPU 2.40 GHz, Dual-Core CPUs) which runs CentOS5.4 operating system, the platform uses the xen-3.0.3-94.el5_4.3 as Hypervisor and Linux kernel 2.6.18-164.15.1.el5.centos.plusxen i686 as Dom0, and uses 1 GBit/s Switched Ethernet. The experiment tests the availability of service during proactive recovery, the resource consumption of replicas domain when creating state checkpoints and replica updates as well as the resource consumption when activate passive replica and suspend active replica.

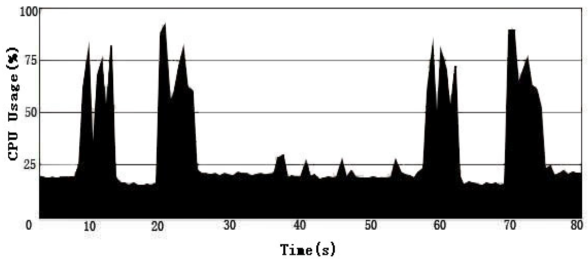

Figure 2. Creating state checkpoint and state updates (both sides peaks); Activating replica and suspending replica (middle peaks)

Fig. 2 shows the system CPU usage when creating state checkpoints, updating replica, activating passive replica and suspending active replica. At about the 10th second in Fig. 2, the system captures the state of replica (using snapshot) and creates the corresponding state checkpoints, and then stores the corresponding state checkpoints in the local. At about the 22nd second, the system updates the replica. During about the 38th second and the 55th second, the system completes activating passive replica and suspending active replica two times. At about the 60th second and the 72nd second, the system separately completes the capture of replica state (using snapshot) and the update of state again.

From Fig. 2 we can clearly see the CPU usage of system in the whole process. When implementing the capture of state, creating the corresponding state checkpoints and updating the state of replica, the CPU usage has some short time peaks, which last about 2s ~ 3s. During this period of time, the maximal CPU usage is between 80%~90%, and the duration time is about tens of milliseconds. While in the time when activating and updating replica, the peaks of CPU usage lasting about 1s~2s, the maximum usage peak is about 25%.Through this experiment we can note that the average CPU usage of system in the whole process keeps between 20%~ 30%, the short time peaks is just about 80%, and the duration time is very short. Thus we can see that the utilization of system resources is not sufficient. In fact, as long as the system memory, disk and network bandwidth and other resources permit, we can create more DomUs to guarantee the full use of system resources, and increase the number of failure replicas that system can tolerate. According to the statistics, the resource utilization of traditional data center server farm is generally lower than 20%, even lower than the lowest 10%. While the virtualization-based cloud computing can significantly improve the resource utilization of platform more than 50% or even higher 80% without reducing service quality. This is the benefit of cloud computing.

Fig. 3 and Fig. 4 respectively show the availability of service when executing proactive recovery based on CC-VIT and traditional BFT. In the experiment, we test the system throughput when implementing proactive recovery in both cases.

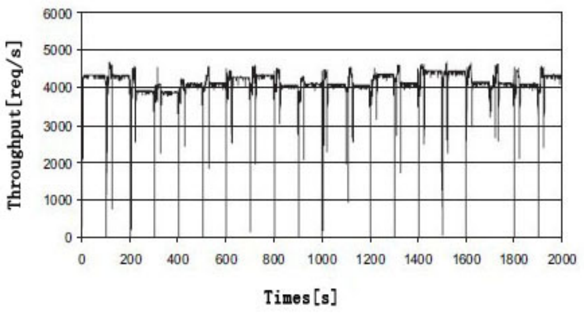

Figure 3. Proactive recovery based on CC-VIT

We adopt the proactive recovery based on CC-VIT model in Fig. 3, and we can see the system throughputs are almost continuous in the whole process. That means the provided system services are almost continuous. During the running of the service, when the faulty replicas are going to break through the threshold F and then proactive recovery is triggered. In this system, in order to reduce the downtime of system services, we adopt a parallel recovery strategy that creates a new domain in parallel to the execution of the other applications. In CC-VIT system we could clone a passive replica to quickly create a new replica domain with the latest available state, and then obtain the state after the last state updates through state transfer. After state updates and state transfer, the replica manager hands over request execution to the former passive replica (new replica), and shuts down the old faulty active replica. In this way, a potentially faulty replica is efficiently and continuously replaced with a clean one. In fact, there will be a short interruption of service during the switch of the new and old replicas, as the system will temporarily suspend the current requests when capturing state (using snapshot) and detaching disk volumes during the state transfer of replica [8]. After this step, the system will re-execute all requests executed by the old replica after the snapshot has been taken (capturing state) to guarantee the continuity and correctness of service. Sometimes, users may not feel this short interruption of system service when in the superior performance server or lower load platform.

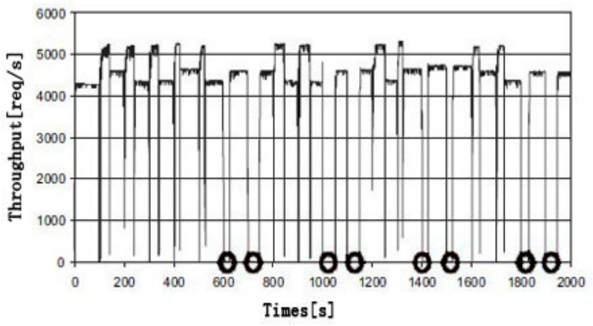

Figure 4. Proactive recovery based on traditional BFT

In order to compare with the proactive recovery based on CC-VIT model, we adopt proactive recovery based on traditional BFT in Fig. 4. In this condition, when the faulty replicas are going to break through the threshold F, the system services will be interrupted, and then proactive recovery is triggered. Before the new replica replaces the old replica, the system services are down all the time. For example, at about the 600th second, the 700th second, the 1000th second, the 1100th second, the 1400th second, the 1500th second, the 1800th second, the 1900th second in Fig. 4, the system successively implements proactive recovery strategy, but the service is obviously break and the maximal interruption of service has reached 50s ~ 60s. Therefore, such a long service interruption will undoubtedly be a fatal killer for cloud computing platform on which the critical business is running. Compared with the two methods, we can note that proactive recovery based on CC-VIT can significantly improve resource utilization and guarantee the system services almost continuous.

VII. Feasibility Validation and Case Analysis of Secret Sharing

A. Validate Property of Reconstruct Secret

According to the scheme designed in section V, $\forall X \notin \beta$, at first we prove $\forall W_i \notin \omega_\beta$, $X \cap W_i \neq \phi$. Assume to the contrary that there exists $W_j \in \omega_\beta$ such that $X \cap W_j = \phi$. Then $X \cap \overline{B_j} = \phi$ according to $W_j = \overline{B_j}$, that is $X \subseteq B_j$. So $X \in \beta$ is in contradiction to the fact that $X \notin \beta$. Therefore, □$W_i \in \omega_\beta$, $X \cap W_i \neq \phi$. By the step 4 of the scheme, each partition K_i of secret K is associated with $W_i \in \omega_\beta$, according to $\forall W_i \in \omega_\beta$, $X \cap W_i \neq \phi$ we should know, participants of X can collect every partition K_i, where$1 \leq i \leq m$. This we can calculate K by $K = \sum_{i=1}^{m} k_i$. So $H(K|\{g(p_i|p_i \in X)\}=0$, that is this scheme satisfies the property of reconstruct secret.

B. Validate Property of Perfect Confidentiality

According to the scheme designed in section V, $\forall X \in \beta$, by the step 1 of this scheme, $\exists B_i \in \beta_{\max}$, $1 \leq i \leq m$, then have $X \subseteq B_i$, that is $\overline{B_i} \subseteq \overline{X}$. Since $W_i = \overline{B_i}$ then $W_i \subseteq \overline{X}$. By the step 4 of this scheme, only the members of W_i own the partition K_i of K, so no participant in X obtains K_i. Set K' as any secret in the secret space, given a partition $\{K_1,\ldots,K_m\}$of K, obviously, $\{K_1,\ldots, K_{i-1}, K_i +(K'-K), K_{i+1}, \ldots,K_m\}$ is a partition of K'. We mark the collected partitions of K and K' by participants p_j as $g_K(p_j)$ and $g_{K'}(p_j)$. According to the step 4 in this scheme, K_i are only distributed to the participants in W_i, so the participants in X can collect the same partitions associated with K and K' according to $W_i \subseteq \overline{X}$. So $H(K|\{g(p_i|\ p_i \in X)\}= H(K'|\{g(p_i|\ p_i \in X)\}$, that is this scheme satisfies the property of perfect confidentiality.

C. Case Analysis of Secret Sharing Based on Virtual Adversary Structure

Assuming an intrusion tolerance system is composed by 9 virtual machines, and it may have a certain kind of system attributes named Attribute which has security risk, and this attribute has four types of attribute values: they are Attribute={a,b,c,d}. Where:

Attribute (1) =…=Attribute (4) =a
Attribute (5) =Attribute (6) =b
Attribute (7) =Attribute (8) =c
Attribute (9) =d

When a certain type of attribute value is exploited by an attacker, all the virtual machines with this attribute value may be successfully intruded by intruders. Assume that we need to design this system, which is able to tolerate at most any two virtual machines are simultaneously corrupted, or is able to tolerate all the virtual machines with a certain type of attribute value are simultaneously corrupted. For this example, we have the following virtual adversary structure: $g(S)= \overline{\Theta_3^9(s)} \vee \overline{\Theta_2^4(x_a(s), x_b(s), x_c(s), x_d(s))}$,where the characteristic function $x_a : 2^P \rightarrow \{0,1\}$ satisfies: $x_a(S)=1 \Leftrightarrow \exists i \in S$, Attribute(i)=a, characteristic function x_b, x_c, x_d and so on.

Table I, Table II and Table III show an example which realizes the scheme of secret sharing based on virtual adversary structure, where the participant set is P={p1,…,p5}, the maximal virtual adversary structure is $\beta_{\max}=\{(p_1\ p_3\ p_4), (p_2\ p_3), (p_5), (p_1\ p_2\ p_4)\}$, corresponding write structure is $\omega_\beta=\{(p_2\ p_5), (p_1\ p_4\ p_5), (p_1\ p_2\ p_3\ p_4), (p_3\ p_5)\}$.

TABLE I.
WRITE STRUCTURE AND PARTITION

Write Structure	Partition
(p_2 p_5)	S_1
(p_1 p_4 p_5)	S_2
(p_1 p_2 p_3 p_4)	S_3
(p_3 p_5)	S_4

TABLE II.
PARTITIONS OF PARTICIPANT

Participant	Partition			
	S_1	S_2	S_3	S_4
p_1		Y	Y	
p_2	Y		Y	
p_3			Y	Y
p_4		Y	Y	
p_5	Y	Y		Y

TABLE III.
MAXIMAL VIRTUAL ADVERSARY STRUCTURE AND PARTITIONS

Maximal Virtual Adversary Structure	Partition Set
(p_1 p_3 p_4)	$\{S_2, S_3, S_4\}$
(p_2 p_3)	$\{S_1, S_3, S_4\}$
(p_5)	$\{S_1, S_2, S_4\}$
(p_1 p_2 p_4)	$\{S_1, S_2, S_3\}$

We can see from Table 3 that an arbitrary participant set in any virtual adversary structure together can not collect all the 4 partitions. So intruders can not obtain any information about secret S; while authorized participants subset ω_β can collect all the 4 partitions to reconstruct secret S. Therefore, this scheme both satisfies the property of perfect confidentiality and reconstruct secret.

VIII. CONCLUSION

This paper constructs a virtualization intrusion tolerance system(CC-VIT system) based on cloud computing by research on the existing virtualization technology, and then presents a method of intrusion tolerance to protect sensitive data in data center of cloud computing based on virtual adversary structure in CC-VIT system platform. This system allows tolerating F faulty replicas in N=2F+1 replicas and ensuring that only F+1 active replicas to execute during the intrusion-free stage, and the remaining replicas are all put into passive mode. While the traditional Byzantine fault-tolerant algorithm tolerates F faulty replicas require N = 3F +1 replicas. Experiment shows that the CC-VIT system can significantly improve resource utilization, guarantees the system service almost continuous, improve the system QoS, and realize fault and intrusion tolerance. Sensitive data protecting method only simply executes modular addition and modular subtraction operations in the stage of the operation without complicated modular exponentiation operation. Therefore this scheme has high computational performance and can not increase the storage load in system. Complementing with each other, these two methods can achieve the double size intrusion tolerance protection in the whole business process of cloud computing.

ACKNOWLEDGMENT

This work has been supported by nature science fund project of Inner Mongolia (No: 20080404ms0903) and by spring sunlight project of ministry of education (No: Z2009-1-01044). Several people in network security and cloud computing areas have contributed to the security of cloud platform, as well as to the related secure virtualization and trusted computing efforts. We would like to thank all the teachers and colleagues in University of Science and Technology Beijing and Inner Mongolia University of Science and Technology, without their help this paper can not come out perfectly. We would also like to thank the Xen community for their support to construct CC-VIT model.

REFERENCES

[1] Peng Liu. The definition and characteristics of cloud computing [EB/OL].(2009-02-15). http://www.chinacloud. cn/ show.aspx ? id=741&cid=17.

[2] Hans P.Reiser,Rüdiger Kapitza. VM-FIT: Supporting Intrusion Tolerance with Virtualization Technology [J]. In Proceedings of the 1st Workshop on Recent Advances on Intrusion-Tolerant Systems, March 23 2007. pp.18-22.

[3] Chu-Hsing Lin, Wei Lee. Efficient Secret Sharing with Access Structures in a Hierarchy. *Proceedings of the 19th International Conference on Advanced Information Networking and Applications,* March 2005, vol.2, pp.123-126.

[4] C. C. Chang, C. H. Lin, W. Lee, and P. C. Hwang. Secret Sharing with Access Structures in a Hierarchy. *International Conference on Advanced Information Networking and Applications 2004 (AINA)*, Japan, Mar 2004, Vol. 2, pp.31-34.

[5] Yuan-Bo Guo, Jian-Feng Ma. Practical Secret Sharing Scheme Realizing Generalized Adversary Structure [J]. *J. Comput. Sci. & Technol,* July 2004, Vol.19, No.4, pp.564-569.

[6] Y.Desmedt, Y.Wang, M.Burmester. A Complete Characterization of Tolerable Adversary structures for secure Point-to-Point Transmissions without Feedback [J]. *16th Annual International Conference*, ISAAC 2005, pp. 277-287.

[7] H.P.Reiser, F.J.Hauck, R.Kapitza,and W.Schrder-Preikschat. Hypervisor-based redundant execution on a single physical host [J]. In Proc. of the 6th European Dependable Computing Conf.,Supplemental Volume-EDCC'06(Oct 18-20,2006,Coimbra,Portugal),2006. 67~68.

[8] Tobias Distler, Rüdiger Kapitza,Hans P.Reiser. Efficient State Transfer for Hypervisor-Based Proactive Recovery

[J].WRAITS'08.2~4,April 1,2008 Glasgow, Scotland Copyright 2008 ACM 978-1-59593-986-9.

[9] C. Cachin and J. A. Poritz. Secure intrusion-tolerant replication on the internet. In Intl. Conf. on Dependable Systems and Networks,pages 167-176, 2002.

[10] M. Castro and B. Liskov. Practical Byzantine fault tolerance. In OSDI '99: Proc. of the 3rd Symp. on Operating Systems Design and Implementation, pages 173–186. USENIX Association, 1999.

[11] Hans P.Reiser, Rüdiger Kapitza.Hypervisor-Based Efficient Proactive Recovery[J]. In Proc. of the 26th IEEE Symposium on Reliable Distributed Systems - SRDS'07 (Oct 10-12, 2007, Beijing, China)，2007.83~92.

[12] Byung-Gon Chun, Petros Maniatis, Scott Shenker. Diverse Replication for Single-Machine Byzantine-Fault Tolerance [J].Proceedings of USENIX Annual Technical Conference, Boston, Massachusetts，June 2008. 287~292 .

[13] Shamir.A .How to share a secret [J].*Communication of the ACM, 1979*, 22 (11), pp.612-613.

[14] Blakley G R. Safeguarding cryptographic keys[C]. *Proceedings of the National Computer Conference.* Montvale: AFIPS Press, 1979, pp.313-317.

Jingyu Wang, Born in Kaifeng, China, July , 1976. Received master degree from in computer engineering Inner Mongolia University of Science and Technology, Baotou, China, June 2002. His research interests include grid and cloud computing security.

Xuefeng Zheng, Born in Fujian, China, January , 1951. Received master degree from University of Science&Technology Beijing, China, June 1982. His research interests include network and cloud computing security.

Dengliang Luo, Born in Chongqing, China, July 6, 1985. Received master degree in computer engineering from Inner Mongolia University of Science and Technology, Baotou, China, June 2011. His research interest is cloud computing security.