A Two-level Application Transparent Checkpointing Scheme in Cloud Computing Environment

He Hui[1,2*], Zhang Zhan[1], Wang Bai-Ling[2], Zuo De-Cheng[1*]
and Yang Xiao-Zong[1]

[1]*School of Computer Science and Technology Harbin Institute of Technology, Harbin, China*
[2]*School of Computer Science and Technology*
Harbin Institute of Technology at Weihai, Weihai, China
[*]*{hehui@hitwh.edu.cn，zdc@hit.edu.cn}*

Abstract.

Cloud computing has been widely applied to a wide variety of computing environments, with the traditional distributed computing environment, cloud computing using virtual machines to achieve dynamic resource partitioning.

Checkpoint recovery technology is a low-cost method to improve the system availability. This paper analyzes the characteristics of cloud computing, virtualization technology and the checkpoint/recovery technology, a mechanism for integration of virtualization technology and checkpoint/recovery technology has been proposed, this mechanism uses coordinated checkpoint protocols at node level, while uses communication-induced protocols at system level. This scheme is transparent to applications with small performance overhead.

Keywords: *Cloud computing, Virtualization, Checkpoint protocols*

1. Introduction

In recent years, cloud computing have been widely used and changed the traditional computing model. Cloud computing aggregates huge amounts of distributed virtualized computers presented as a huge resource pool, uses service-level agreements providing service to consumers [1, 15]. Most cloud computing platforms use system level virtualization techniques for resources management, which promoted system utilization, and reduced deployment time, supplying on demand, *etc*.

Since more and more traditional distributed systems have migrated to cloud computing platforms, increasingly virtual machines (VMs) have been used instead of physical machines. Runtime errors are more likely to happen because systems become larger and larger.

Current techniques of fault toleration mainly have two mechanisms: proactive schemes and reactive schemes. Nagarajan promoted a method migrate unhealthy node to healthy ones automatically before faults happen [2]. Riser uses hypervisor-based technology realized this mechanism minimizes the time for proactive reboot interferes system availability [3].

Checkpoint and rollback techniques are widely used as reactive schemes. There are three checkpointing protocols: The last checkpoint of each process has been always the consistent global checkpoint in coordinated checkpointing protocols, while the coordinating times are intolerable when the system become larger and larger [4]. In uncoordinated checkpointing protocols, processes can make checkpoints autonomously without times of coordinating. However, domino effect may happen leading to huge rollback overhead [5]. Communication-Induced checkpointing protocols piggyback some of control information in messages to achieve synchronization of all processes without suspending all processes for coordination [6].

In system level virtualization, virtual machine monitors (VMMs) or hypervisor provides such checkpoint schemes [7]. Checkpointing in VMMs level has many advantages: (I) no need to modify application codes or operating system kernels; and (ii) provides a whole VM checkpoint, both applications and the operation system states. However, checkpointing in VMMs needs more disk space than application checkpoints, leads to more unavailable time of the whole system while saving the VM states, can't meet the largest distributed computing demand.

This paper presents an efficient checkpointing scheme in cloud computing environments at system-level virtualization layer, using coordinated checkpointing protocols at node level, and Communication-Induced checkpointing protocols at the system level. Advantages of this scheme include:

(i) Controllable synchronous overhead. Since there are several virtual machines on each node, every node synchronize separately, avoids the whole system coordinating.

(ii) Prevention of disk accessing competition. Shared storages are commonly used in cloud environments, storages accessing competition would be happen when huge amounts of VM saving their states simultaneously. For communication-induced checkpoint protocol is used at node level, they access shared storages asynchronously.

(iii) Light-weight checkpointing in node level. Taking entire snapshots of a VM may not be too huge to acceptable, but in cloud computing, massive of VMs snapshots would be terrible to saving the whole system states. Checkpointing node VMs in increment method using cache like checkpoint method is introduced to reduce checkpoint overhead.

2. Related Work

This paper [8] focus on fast memory state synchronization using virtualization for fault tolerance. Three optimization techniques are described: fine-grained dirty region tracking, speculative state transfer and synchronization traffic reduction using active slave.

Remus [9] describes providing high availability service without modifying existing software to be protected from physical machine failure, with only seconds of downtime for transparently restart on alternate host in face of running host failure.

VM-μCheckpoint proposes a lightweight mechanism for VM checkpointing and recovery[10].To minimize the checkpoint overhead, VM-μCheckpoint save incremental checkpoints in volatile memory with COW and dirty-page prediction mechanism .This mechanism uses in-place recovery to reduce recovery time.

VirtCFT [11] builds on system-level virtualization , uses coordinates distributed checkpointing save the whole virtual cluster including the status of CPU, memory, disk and the network communications periodically, and provides a transparent fault tolerant platform.

BlobCR [12] uses two stage checkpointing, the first stage running inside the guest operating system, the second stage running at VMMs level. Through this mechanism, disk space and overhead of checkpoint-restart are obviously decreasing.

3. System Model

Assume a large amount of processes are deployed on M virtual machines running at N nodes (physical machines), each vm runs one process at the moment. These processes communicate through messages, the delays of messages transmission is finite and arbitrary. These processes are in accordance with fail-stop model [13].

Let send(m) denote the send event of message m, and deliver(m) denote the deliver event of message m.

Let $\xrightarrow{hb}$ denote Lamport's happen before relation [14].

Suppose C is a set of local checkpoints, each element comes from one process, C is called consistent global checkpoint when there haven't two checkpoints that one checkpoint happened before another one [16, 17].

4. System Overview

Figure 1 shows system deployment at a node in cloud computing environments. This scheme implements on Xen VMM level, running at Dom0.

The coordinated CR module is responsible for node-level checkpointing. This module will make a node checkpoint when receiving a checkpoint request from communication-induced CR module, without regard to outgoing and ingoing communications.

The communication-induced CR module is responsible for system-level checkpointing, use communication-induced checkpoint and recovery protocols to achieve small system suspending time. This module sends normal or forced checkpoint request to coordinate CR module.

Communication module deals with the communication of guest OS, messages among the node cannot lead to a forced checkpoint request. When received a message from the other node communication module will send a message to communication-induced CR module for judgment whether take a forced checkpoint or not.

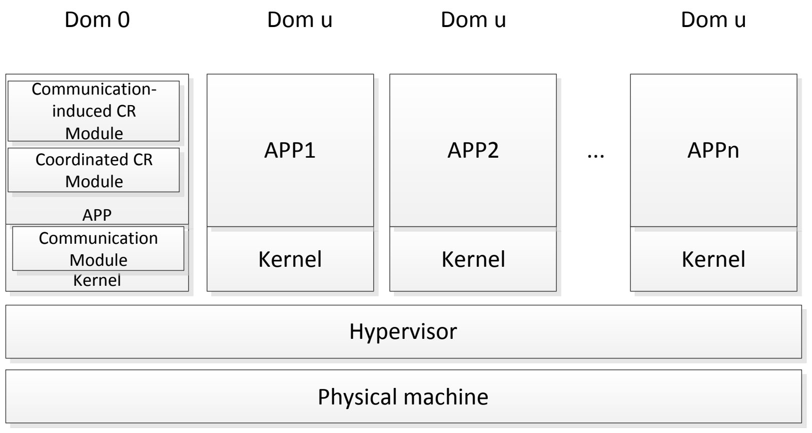


Figure 1. Deployment of the Checkpoint-Recovery System

4.1. Implement of the Coordinated CR Module

Unlike traditional coordinated checkpoint-recovery protocols, this paper propose a simple scheme for virtual machines running on the same node, controlled by the same hypervisor and Dom0.

To avoid whole VM snapshot, this paper describes a light-weight in the memory increment scheme. The i'th of Checkpoint of each VM contains four parts: $<reg_i>,<inm_i>,<ind_i>,<com_i>$.

reg_i : The register states of the VM.

inm_i : Increment memory status of the VM.

ind_i : Increment disk status of the VM.

com_i : Communication state of the VM.

The coordinated CR module works as follow.

Coordinate CR module suspend all VM on the node simultaneously.

Save register states to all VMs checkpoints.
Save increment memory status to checkpoints.
Save increment file to checkpoints, which provided by stackable file system, *e.g.*, unionFS.
Save communication channels to checkpoints.
Send success message to communication-induced CR module.
Resume all VMs to running states.

To decrease the overhead of checkpointing memory states, this paper proposed a novel cache like checkpoint scheme. The coordinated module keeps a pointer array of each VM's dirty pagers; the elements of the array are a pointer indicating dirty pages since the last checkpoint. The size of the array should be appropriate, and this array only records the most recently modified dirty page's address. Since amounts of dirty pages will lager than this array size, the recent less used pointer should be replaced out, and coordinated module will save this dirty page to shared storage accordingly. So, during each checkpoint interval, coordinated module only needs to records the dirty page addresses in the array, and save the pointed dirty pages by the element that need replaced out of the array. At the time of checkpointing, this module only needs to save dirty pages pointed by the array, this method decreased checkpoint overhead obviously. Figure 2 shows the main idea of the memory checkpoint schme.

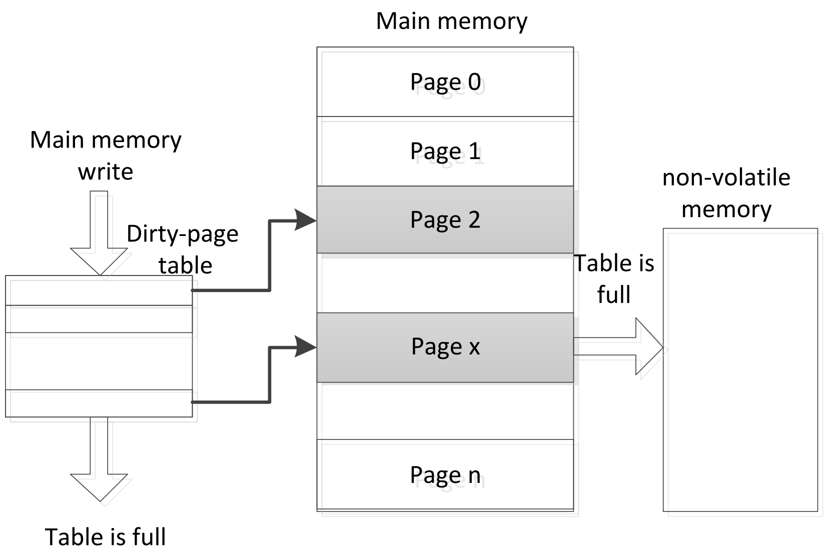


Figure 2. Low-overhead Memory Chekpointing Schme

4.2. Implement of the Communication-induced CR Module

Assume the whole system have N nodes, corresponding VM numbers on nodes are $\{X_1, \ldots, X_N\}$.

$P_{n_m,k}$: refers to the k'th VM checkpoint of VM m on node n.

$S_{i,sn}$: set of the sn'th checkpoint of node i.

$S_{i,k} = \{P_{i_1,k}, P_{i_2,k}, \ldots, P_{i_{Xi},k}\}$

L_k : set of the k'th checkpoint from every VM in the whole system.

$$L_k = \{S_{1,k},...,S_{N,k}\} = \{\{P_{1_1,k},...,P_{1_{X1},k}\},...,\{P_{N_1,k},...,P_{N_{XN},k}\}\}$$

Each communication-induced CR module running on the node of cloud platform uses the traditional index-based checkpoint protocols (BCS) [12] as follow.

```
Basic checkpoint:
    When time to basic checkpoint period:
        $sn_i \leftarrow sn_i + 1$
        Send a checkpoint request to coordinate CR module;
        A basic node checkpoint is taken when received the success message from the
        coordinated CR module;

Force checkpoint:
    When received a message from another node
        If $sn_i < m.sn$
        Then $sn_i \leftarrow m.sn$
            Send a checkpoint request to coordinate CR module;
        The message is processed;
```

5. Proof of Global Consistency

System level based virtualization provides VMs communication with asynchronous IO ring scheme, this method makes the processing of the messages of the ring will prolong until checkpointing have finished. Figure 3 shows that message m_1 should have been processed before time tb if no checkpoint happens, like m_1'.

At the time t_a the node started checkpointing, and finished at time t_b. Message m_1 was processed after checkpoint finished. P_{11}, P_{12}, P_{13} are three processes running on node 1, while P_{21}, P_{22} running on node 2. $P_{11,0}$ notes the first corresponding VM checkpointing.

Although without adopting the traditional two stage coordinated scheme, the checkpoints on same node should be consistent as well. Because these VM takes a checkpoint simultaneously, no orphan message will happen, which leads to partial consistent checkpoint on one node. The under processing messages, include inter node messages like, ingoing messages like m_2 and outgoing messages like m_3, did not break the node checkpointing consistency.

Index-based checkpoint protocol is used to maintain a global consistent node checkpoint, as L_1 shown in Figure 3. Then all VM checkpoints in the system from the node level recovery line will be globally consistent.

Proof: Assume L_k is a set of node consistent checkpoint taken according the scheme above. Accordingly, $\neg\exists S_{m,k}, S_{n,k}$ satisfying

$$S_{m,k} \in L_k,\ S_{n,k} \in L_k \text{ and } S_{m,k} \xrightarrow{hb} S_{n,k} \vee S_{n,k} \xrightarrow{hb} S_{m,k}.$$

Presumes L_k is not a global consistent VM checkpoint set. So there must have $\exists P_{m_r,k} \in L_k, P_{n_s,k} \in L_k$, satisfying $P_{m_r,k} \xrightarrow{hb} P_{n_s,k}$, this implies a Z-path from $P_{m_r,k}$ to $P_{n_s,k}$. Since communication inside node could not cause the Z-path come into being for using coordinating checkpoint protocols, so the Z-path can only bring out by node communication, which implies $S_{m,k} \xrightarrow{hb} S_{n,k}$, the assumption does not hold for analysis above.

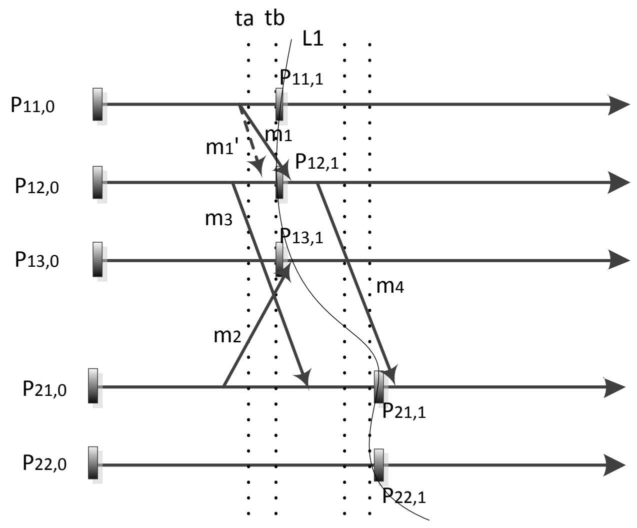


Figure 3. Consistent Analysis of Node Checkpoint and VM Checkpoint

6. Performance Analyses

Unlike traditional two stages coordinated checkpoint scheme, Dom0 and hypervisor have privilege to access the resources of guest OS, all virtual machines' checkpoint can be taken in Dom0. The time cost of node checkpointing T_n consists of several parts: T_s, the time of suspend; T_a, the time of saving registers of VM; T_m, the time of saving cached dirty pages in memory; T_d, the time of saving incrementing files, T_c, times of saving communication information, T_r, time of restart VMs to run.

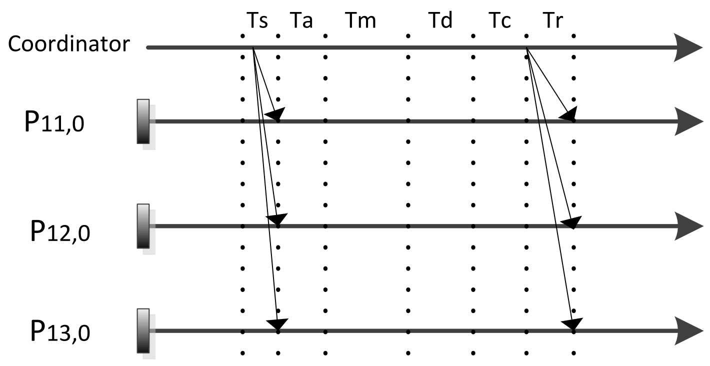


Figure 4. Time Cost of Node Checkpointing

So total overhead of checkpointing of the system is:

$$T_{overhead} = \sum_{i=1}^{N} \left(N_{i_{basic}} + N_{i_{force}} \right) T_{i_n} \qquad (1)$$

Synchronization overhead only happens at node level, when one node is checkpointing, the other nodes running normally. This scheme eliminates system availability caused by using coordinated checkpoint-recovery protocols of the whole system.

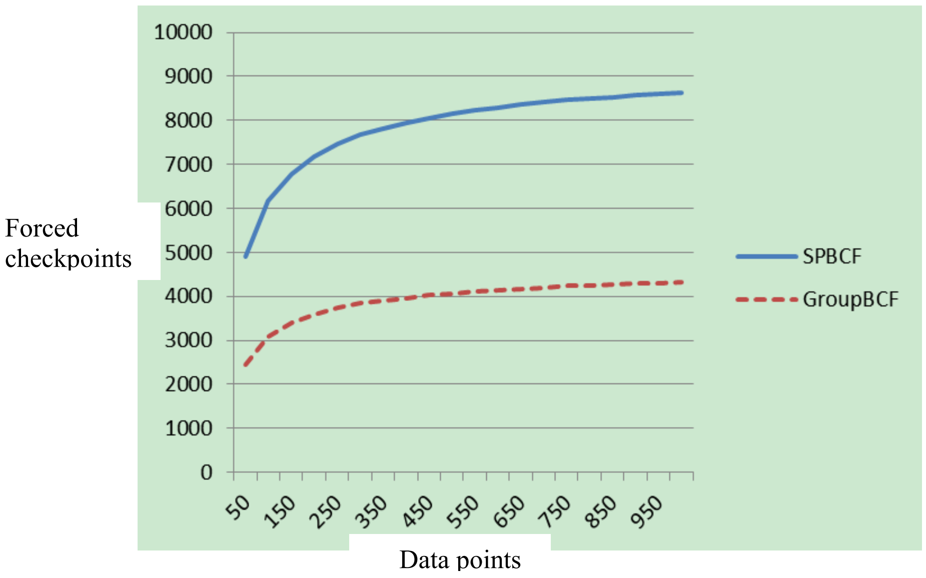


Figure 5. Comparion of Forced Checkpoints

Simulation is formed as a system up to 1000 VMs, the max numbers on one node is 8 VMs. Totally 10000 events happened in this system, and basic checkpoints per VM is 100, Figure 5 shows that the comparison of forced VM checkpoints while using two-level mechanism. The solid line indicates using BCF protocol at system level for symmetric processes, because communication between nodes is very frequent, and each node-level forced checkpoint consist 8 VM checkpoints, so the forced checkpoints are huge when system become larger and larger. The broken line reflects the forced checkpoints that every VM prefers to communicate with VMs on the same node, in this simulation, the groupBCF presumes the probability of communication within the node is 50%. The result shows that, this mixed scheme is suited for the applications that communicate locally frequently.

The disadvantage of this checkpoint-restart framework in this paper is increased the additional computing overhead to Dom0, which may lead to the risk of node unavailable when Dom0 failure happens.

7. Conclusions

In this paper, communication-induced checkpoint-restart mechanism is used at node level, thus eliminates unavailability leading from synchronization in coordinated checkpoint-restart systems. Synchronization overhead only exists in every physical node in this paper, because several virtual machines running on each node, moreover since cached increment memory checkpointing mechanism is adapted, the synchronization overhead is very small, suitable for large-scale deployment of distributed systems.

Acknowledgements

Supported by the National Science Naturel Foundation of China No 61003047, Supported by the National Science Nature Foundation of China Grant No 61170262.

References

[1] R. Buyya, C. S. Yeo and S. Venugopal, “Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities”, in High Performance Computing and Communications, 2008, HPCC'08. 10th IEEE International Conference, **(2008)**.

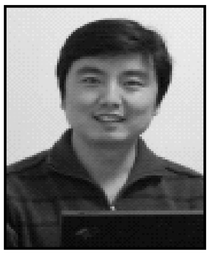
[2] A. B. Nagarajan, F. Mueller, C. Engelmann and S. L. Scott, "Proactive fault tolerance for HPC with Xen virtualization", Proceedings of the 21st annual international conference on Supercomputing, ACM, **(2007)**.
[3] H. P. Reiser and R. Kapitza, "Hypervisor-based efficient proactive recovery", Reliable Distributed Systems, 26th IEEE International Symposium on IEEE, **(2007)**.
[4] G. Cao and M. Singhal, "On coordinated checkpointing in distributed systems", Parallel and Distributed Systems, IEEE Transactions on, vol. 9, no. 12, **(1998)**.
[5] Y. M. Wang, P. Y. Chung, I. J. Lin and W. K. Fuchs, "Checkpoint space reclamation for uncoordinated checkpointing in message-passing systems", Parallel and Distributed Systems, IEEE Transactions on, vol. 6, **(1995)**, pp. 546-554.
[6] L. Alvisi, E. Elnozahy, S. Rao, S. A. Husain and A. De Mel, "An analysis of communication induced checkpointing", Fault-Tolerant Computing, 1999, Digest of Papers, Twenty-Ninth Annual International Symposium, IEEE, **(1999)**.
[7] G. Vallee, "Checkpoint/restart of virtual machines based on Xen", Proceedings of the High Availability and Performace Computing Workshop (HAPCW 2006), Santa Fe, New Mexico, USA, **(2006)**.
[8] Lu, Maohua and T. -c. Chiueh, "Fast memory state synchronization for virtualization-based fault tolerance", Dependable Systems & Networks, 2009, DSN'09, IEEE/IFIP International Conference, IEEE, **(2009)**.
[9] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson and A. Warfield, "Remus: High availability via asynchronous virtual machine replication", Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, **(2008)**.
[10] Wang, Long, "Checkpointing virtual machines against transient errors", On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International, IEEE, **(2010)**.
[11] M. Zhang, "VirtCFT: A Transparent VM-Level Fault-Tolerant System for Virtual Clusters", Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference, IEEE, **(2010)**.
[12] D. Briatico, A. Ciuffoletti and L. Simoncini, "A DistributedDomino-Effect Free Recovery Algorithm", Proc. IEEE Int'l Symp.Reliability Distributed Software and Database, **(1984)**.
[13] R. D. Schlichting and F. B. Schneider, "Fail-Stop Processors: An Approach to Designing Fault-Tolerant Computing Systems", ACM Trans. Computer Systems, vol. 1, no. 3, **(1983)**.
[14] L. Lamport, "Time, clocks, and the ordering of events in a distributed system", Communications of the ACM, vol. 21, no. 7, **(1978)**.
[15] H. Kim, "A Trust Evaluation Model for QoS Guarantee in Cloud Systems", International Journal of Grid and Distributed Computing, vol. 3, no. 1, **(2010)**.
[16] G. Belalem, L. Allal and C. Dad, "Consistency Management of Replicas in Wireless Grid Environment", International Journal of Grid and Distributed Computing, vol. 3, no. 4, **(2010)**, pp. 33-44.
[17] J. Acosta-Elias, E. Stevens-Navarro and U. Pineda-Rico, "Oriented Consistency Algorithms for Virtual Organizations in Intensive Data Grids", International Journal of Grid and Distributed Computing, vol. 1, no. 1, **(2008)**, pp. 31-38.

Authors

He Hui is working for Harbin Institute of Technology at Weihai as an engineer. His research is mainly on dependable computing, embedded computing.

Zhang Zhan is working for Harbin Institute of Technology as an associate professor. He got the Ph.D. degree form HIT. His research is mainly on dependable computing, mobile computing.

Zuo Decheng is working for Harbin Institute of Technology as a professor. He got the Ph.D. degree form HIT. His research is mainly on dependable computing, mobile computing.



Wang Bailing is working for Harbin Institute of Technology (abstract as HIT) as an associate professor. He got the Ph.D. degree from HIT in 2006. His research is mainly on information security, network security, parallel computing.

Yang Xiaozong is working for Harbin Institute of Technology as a professor. His research is mainly on dependable computing, mobile computing.