HotRestore: A Fast Restore System for Virtual Machine Cluster

Lei Cui, Jianxin Li, Tianyu Wo, Bo Li, Renyu Yang, Yingjie Cao and Jinpeng Huai

ACT lab, Beihang University

2014-11-12

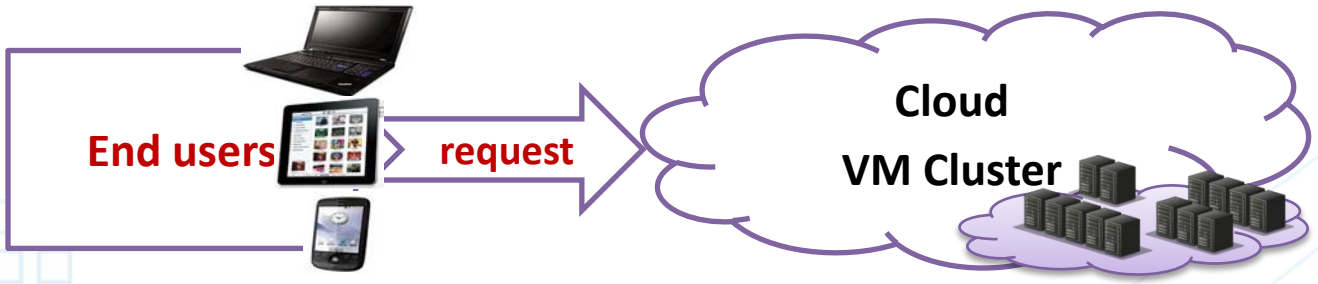
Outline

- Background
- Problems
- Solution & Implementation
- Experimental Results
- Conclusions

The Institute of Advanced Computing Technology

Background

- **Virtual Machine Cluster**
  - Key computing paradigm in cloud
  - Powerful capacity, isolation, scalability
  - Scientific computing, distributed database, web service, etc



vmware

amazon web services™

rackspace HOSTING

Background

- **Failures become common nowadays**
  - Ten thousands of commodity devices

	Annual failure rate	Reference
Computer node	20~60% per processor	J. Physics'07
Storage node	2%~4%, some 3.9%~8.3%	OSDI'10
Network node	1.1%~11.4%	SIGCOMM'11

- **Snapshot & restore**
  - Save the running state, and restore the system from the immediate state upon failures
  - One VM failure leads to the restoration of the whole VMC

Background

- **Failures become common nowadays**
  - Ten thousands of commodity devices

	Annual failure rate	Reference
Computer node	20~60% per processor	J. Physics'07
Storage node	2%~4%, some 3.9%~8.3%	OSDI'10
Network node	1.1%~11.4%	SIGCOMM'11

- **Snapshot & restore**
  - Save the running state, and restore the system from the immediate state upon failures
  - One VM failure leads to the restoration of the whole VMC

VMC restoration occurs frequently to survive from the failures

Outline

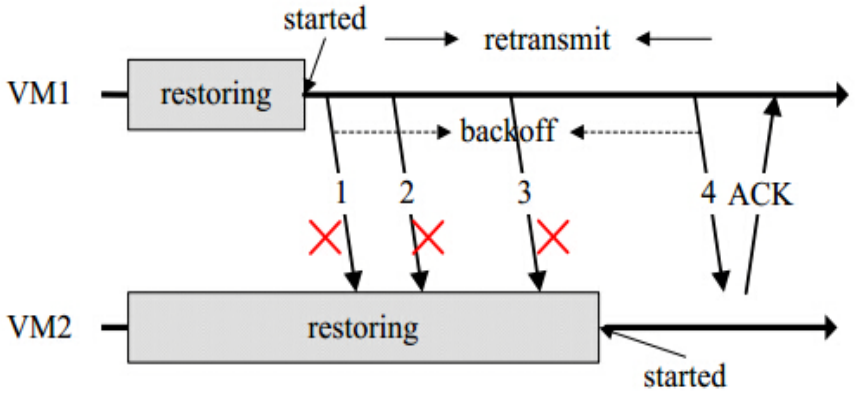
- Background
- Problems
- Solution & Implementation
- Experimental Results
- Conclusions

The Institute of Advanced Computing Technology

Problems

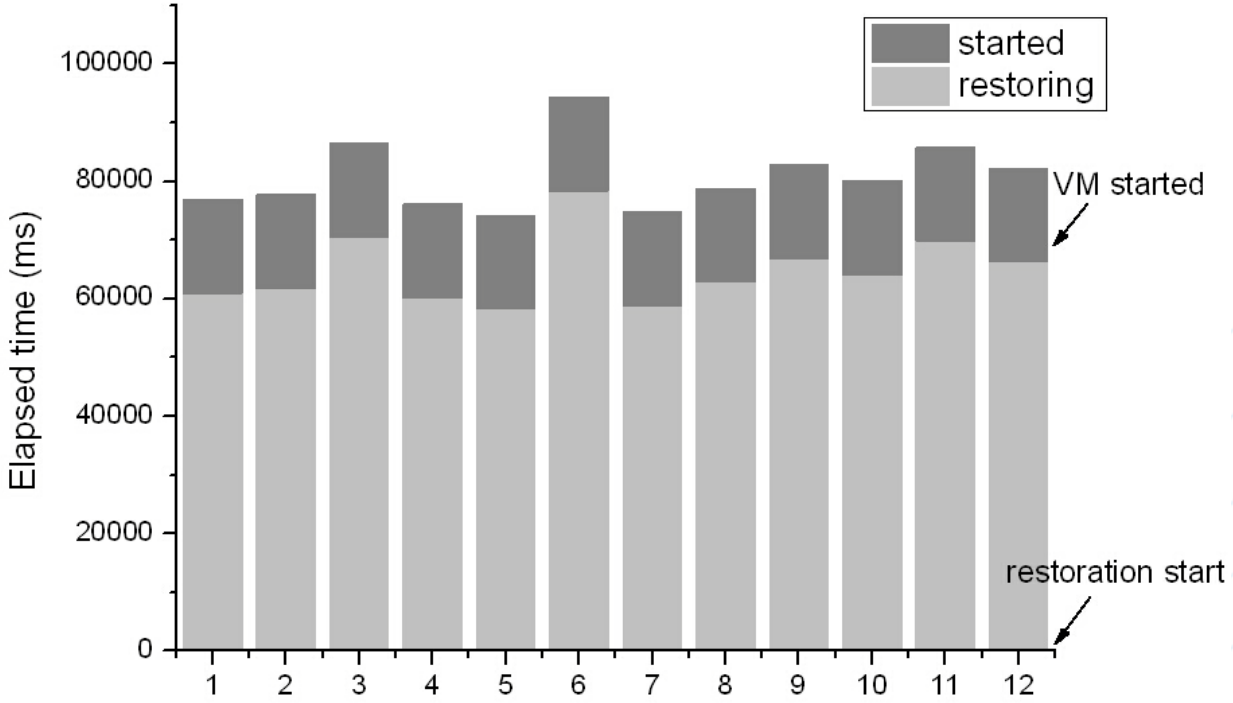
- **Single VM restoration**
  - Retrieve the entire memory state, may be dozens of GBs
  - Long latency to resume the VM, minutes
- **Cluster restoration**
  - Latencies of VMs are various
    - **Heterogeneity, varieties of workloads**
  - Network interruption
    - **TCP backoff.**

vm_1 cannot communicate vm_2 since vm_2 is restoring



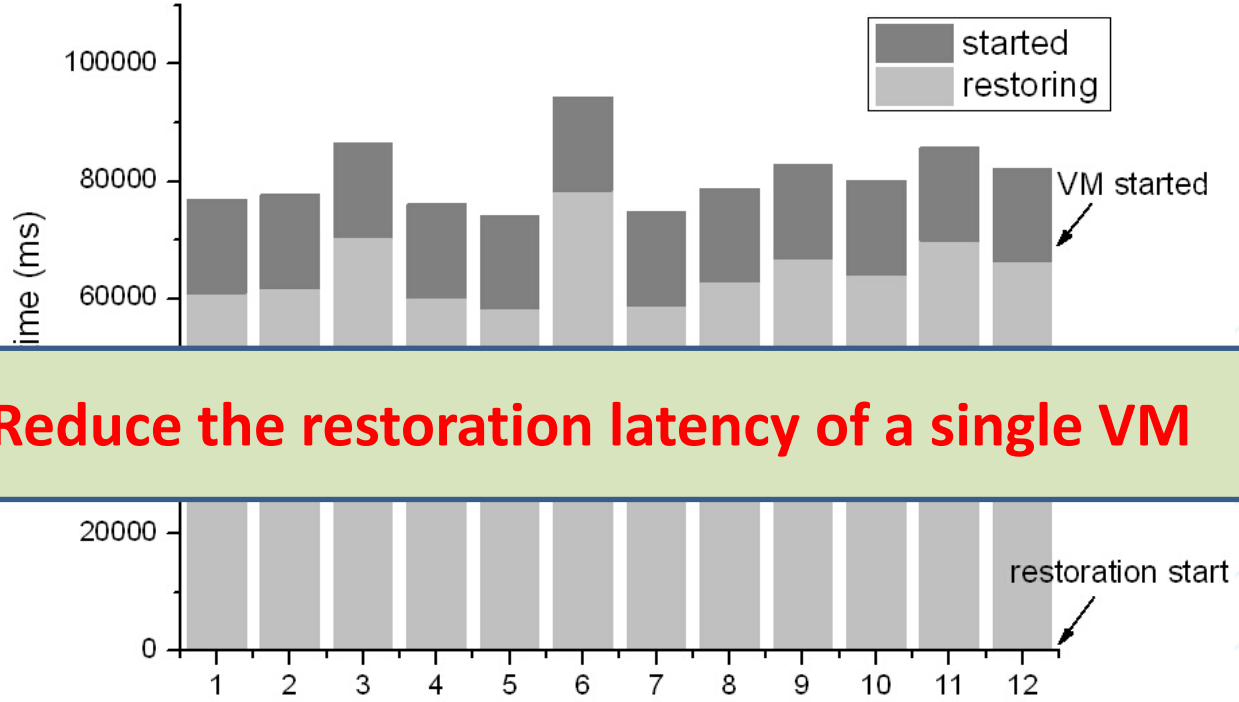
Problems

- **Experimental result**
  - 12 2G memory VMs. Distcc to compile the Linux kernel 2.6.32-5
  - VM_6 is Distcc client, TCP-backoff of VM_6 and VM_7 is 19.6s
  - Distcc would not work until VM6 starts



Problems

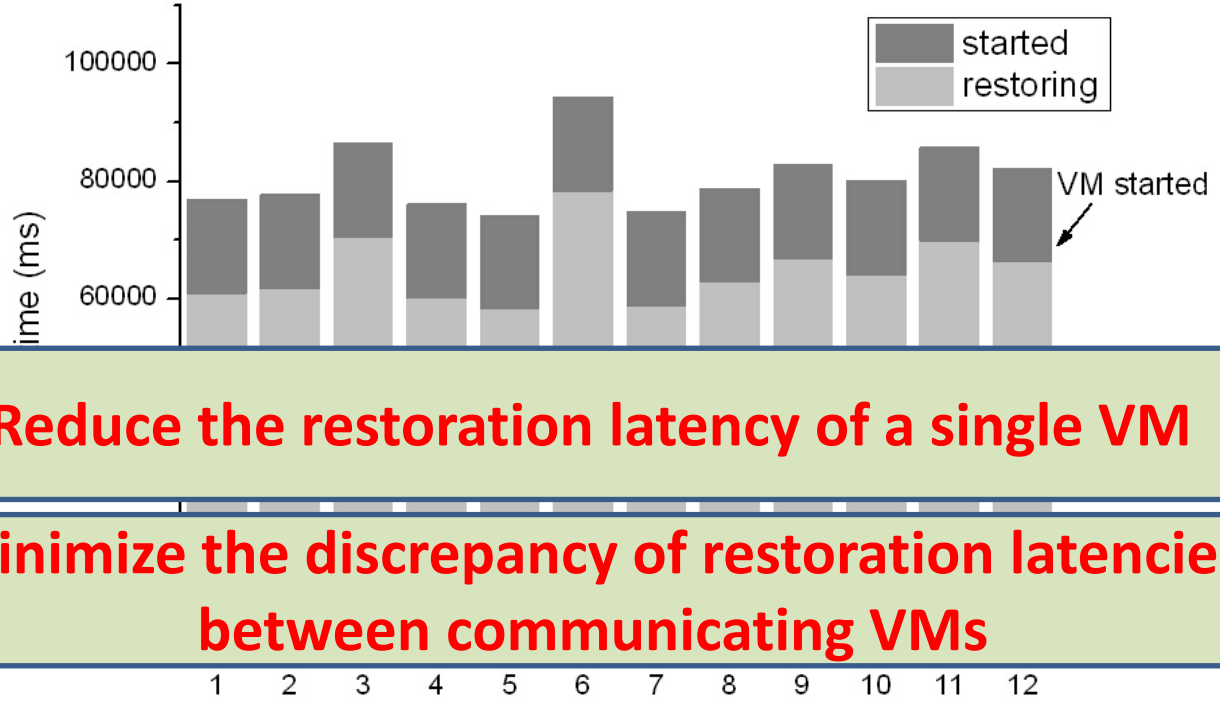
- **Experimental result**
  - 12 2G memory VMs. Distcc to compile the Linux kernel 2.6.32-5
  - VM_6 is Distcc client, TCP-backoff of VM_6 and VM_7 is 19.6s
  - Distcc would not work until VM6 starts



Reduce the restoration latency of a single VM

Problems

- **Experimental result**
  - 12 2G memory VMs. Distcc to compile the Linux kernel 2.6.32-5
  - VM_6 is Distcc client, TCP-backoff of VM_6 and VM_7 is 19.6s
  - Distcc would not work until VM6 starts



Reduce the restoration latency of a single VM

Minimize the discrepancy of restoration latencies between communicating VMs

Outline

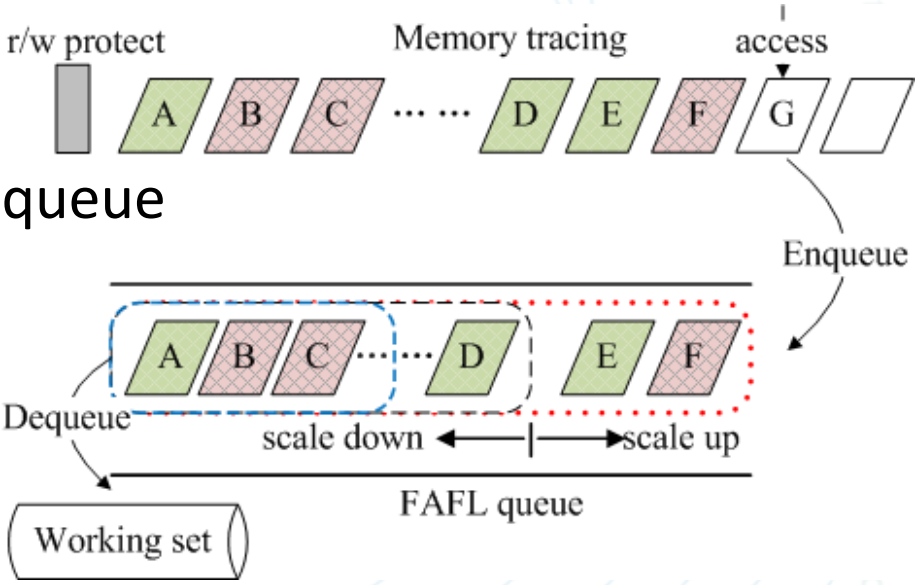
- Background
- Problems
- Solution & Implementation
- Experimental Results
- Conclusions

Solution - Elastic working set

- **Motivation**
  - VM re-execute instructions from the checkpoint state after rollback-recovery
    - The touched pages during checkpoint would be touched again
    - The prior touched pages would be touched preferentially
  - Memory access locality
    - The touched pages take a little fraction of the entire memory state.
- **Working set**
  - Trace memory operation during checkpointing
  - Treat touched pages as working set candidates
  - Load working set rather than the entire memory

Solution - Elastic working set

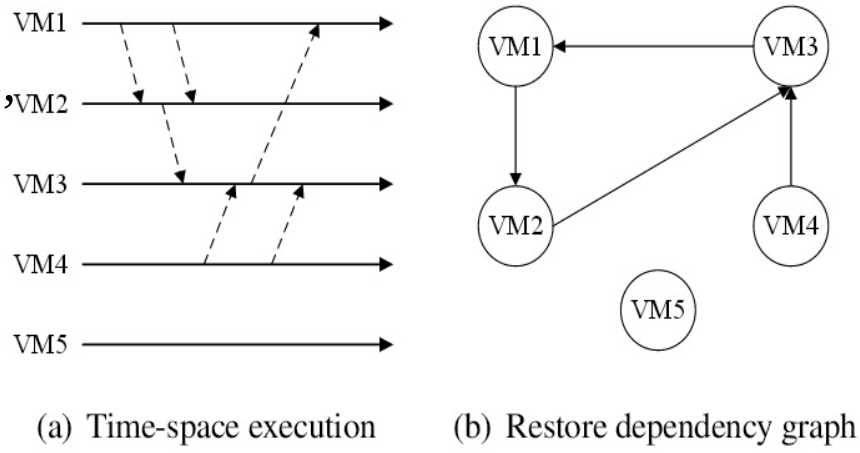
- **How to trace**
  - Post-copy based snapshot
    - **Set read/write protection flag of ptes**
    - **Copy-on-write**
    - **Record-on-access**
  - First access first load (FAFL) queue



- **Elastic**
  - Scale up/down
  - Working set size change on demand

Solution – Restore line

- **Restore line**
  - Arrange the start order of VMs
  - Basic idea:
    - If the receiver starts before the sender, then network interruption disappears.
    - Communication-induced causality
- **Restore dependency graph**
  - If A sends a packet to B, then A->B, and B should start before A.
  - Dependency is transitive
    - A->B, B->C, then A->C
  - Ring is allowed
- **Calculation of restore line**
  - If A->B, B starts first
  - If A, B are in a ring, they start simultaneously
  - Orphan node start freely

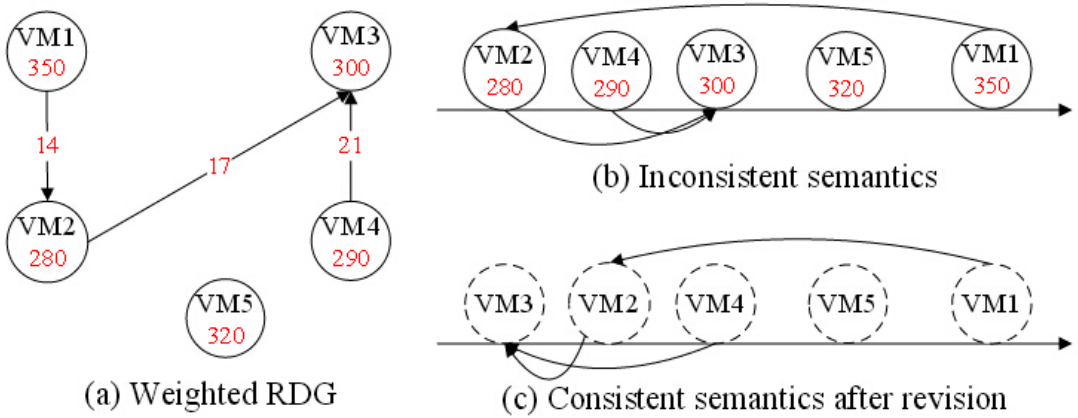


(a) Time-space execution (b) Restore dependency graph

Solution – WSS revision

- **Inconsistency**
  - Order of restore line, and order of working set sizes
    - **A->B in restore line, but A starts before B since WSSa < WSSb.**

Edge: packets cnt
Node: the WSS



- **Revision**
  - $S=\{S_1, S_2, \ldots, S_n\}$ is the previous WSS, $W=\{W_{i,j}|VM_i->VM_j\}$, S* is the revised WSS.
  - Goals
    - **Causality in the restore line**
    - **Minimum modification**

$$\begin{aligned} \min \quad & \textstyle\sum_{i-0}^{n} |\, S_i^* - S_i \,| \\ s.t. \quad & S_i^* - S_j^* \geqslant W_{i,j} \quad (W_{i,j} \in W) \\ & S_i^* - S_j^* \quad - 0 \quad (VM_i \leftrightarrow VM_j) \end{aligned}$$

Implementation details

- **QEMU/KVM platform**
  - VMM layer, no modification of guest OS
- **TCP and UDP packets to build RDG**
  - Intercept packets during checkpoint
  - Src and dst are the VMs within the VMC
- **Intercept and replay the packets**
  - Make the communication after restoration be deterministic

Outline

- Background
- Problems
- Solution & Implementation
- **Experimental Results**
- **Conclusions**

The Institute of Advanced Computing Technology

Experimental results

- **Working set evaluation**
  - **VM Setup:** 2GB RAM, 1 vcpu, 1Gb nic.
  - **Hit rate**: FAFL performs best
  - **Size:** **93.8%** reduction by native approach in QEMU/KVM

Workloads	FAFL	LRU	CLOCK
Gzip	0.806	0.768	0. 883
MySQL	0.947	0.655	0.912
Mummer	0.931	0.835	0.812
Pi	0.628	0.562	0.589
MPlayer	0.890	0.825	0.862

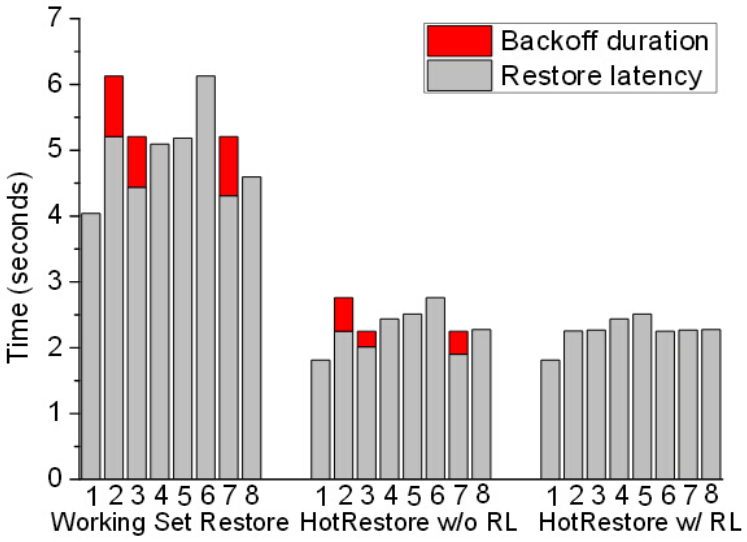
Table 1. Hit rate under various workloads

Workloads	HotRestore	Native
Gzip	61	1052
MySQL	42	1284
Mummer	347	1635
Pi	1.5	736
MPlayer	37	1367

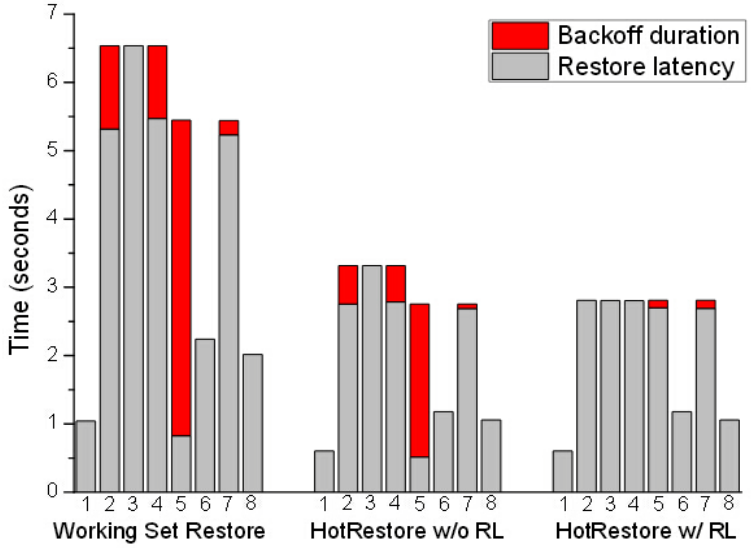
Table 2. Size of loaded data upon restoration (MB)

Experimental results

- **Network interruption (Backoff duration)**
  - **Setup:** 8 VMs, 2GB RAM
    - **Distcc:** client/server
    - **Elasticsearch:** de-centralized
  - **Results:**
    - Latency is reduced, backoff is eliminated under Distcc
    - Backoff is serious under Elasticsearch, HotRestore reduces the duration to sub-seconds



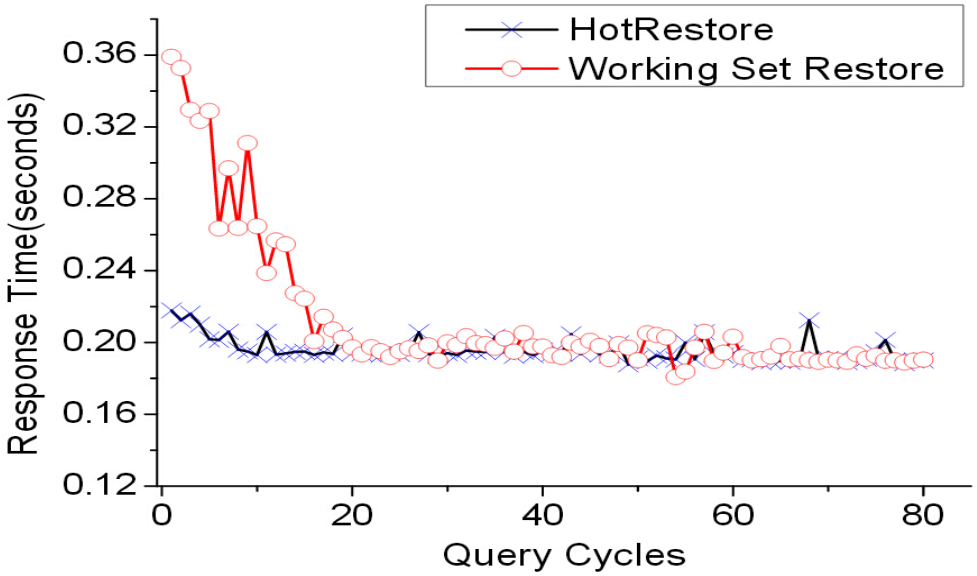
(a) Disruption under Distcc



(b) Disruption under Elasticsearch

Experimental results

- **Application Performance**
  - **Setup:**
    - 8 VMs, 2GB RAM
    - Elasticsearch, ten client query blogs concurrently
  - **Results:**
    - With HotRestore, Elasticsearch server regains the full capacity immediately, while it requires about **6** seconds with Working Set Restore.



Outline

- Background
- Problems
- Solution & Implementation
- Experimental Results
- **Conclusions**

The Institute of Advanced Computing Technology

Conclusions

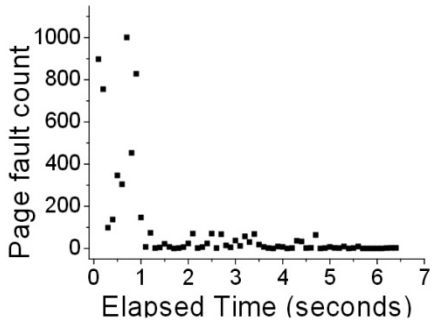
- **HotRestore**
  - Elastic working set
    - **Reduce restoration latency**
  - RDG based restore line
    - **Minimize the discrepancy of restoration latencies on the basis of causality.**
- **Experimental results**
  - Single VM restoration latency, a few seconds
  - 16 VMs, TCP-backoff duration < 1s
  - The VMC resume within a few seconds rather than minutes.
- **Future work**
  - Evaluate HotRestore on SMP VMs
  - Profile the overall performance when multiple snapshots and one restoration are conducted.

Q&A

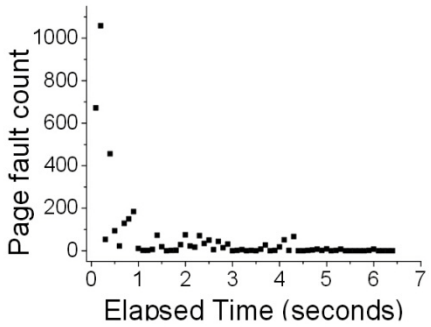
The Institute of Advanced Computing Technology

Experimental results

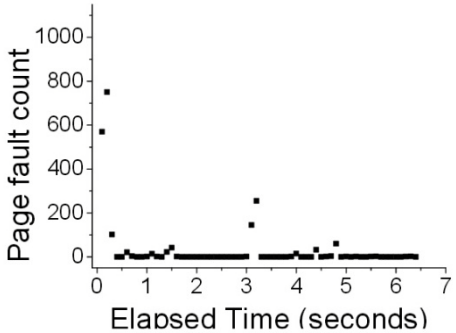
- **Working set scalability**
  - **Setup:** Compilation, trace page faults after restoration in 100ms interval.
  - **WSS:** 18327 pages
  - **Results:** scale up will trigger less page faults, but the amount is little compared to the benefit of less restoration latency



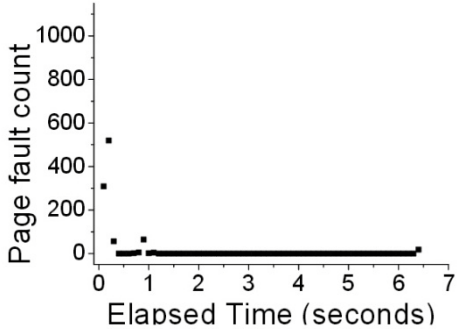
(a) $0.5WSS$



(b) $0.7WSS$



(c) $1WSS$



(d) $2WSS$

	0.5WSS	0.7WSS	1WSS	2WSS
Loaded page	9163	12829	18327	36654
Page faults	5690	3539	2046	958

Experimental results

- **Network interruption (Backoff duration)**
  - **Setup:** 8, 12, 16 VMs, 2GB RAM, Elasticsearch
  - **Results:**
    - Native restore incurs **dozens** of seconds backoff duration
    - Working set restore incurs 2.66 seconds, but the maximum duration reaches **10** seconds.
    - HotRestore reduces the average duration to **0.07** seconds, even the maximum is **0.14** seconds.



(a) 8 VMs (b) 12 VMs (c) 16 VMs

Experimental results

- **Performance overhead**
  - Setup: VM: 2G RAM
  - Results:
    - Increase on snapshot duration is less, e.g., **1.14** seconds on average

Workloads	Baseline	HotRestore
Compilation	85.3	86.6
Gzip	79.5	81.1
Pi	54.2	54.4
Mplayer	72.5	74.2
MySQL	77.3	78.2

Table 1. Snapshot duration

The Institute of Advanced Computing Technology