Live Migration of Virtual Machine Based on Full System Trace and Replay

Haikun Liu, Hai Jin, Xiaofei Liao,
Liting Hu, Chen Yu

School of Computer Science and Technology

Huazhong University of Science and Technology

Services Computing Technology and System Lab (SCTS)
Cluster and Grid Computing Lab (CGCL)

Outline

- Introduction
- Deterministic replay with execution trace
- System design and implementation
- Performance Evaluation
- Conclusion

Live VM Migration

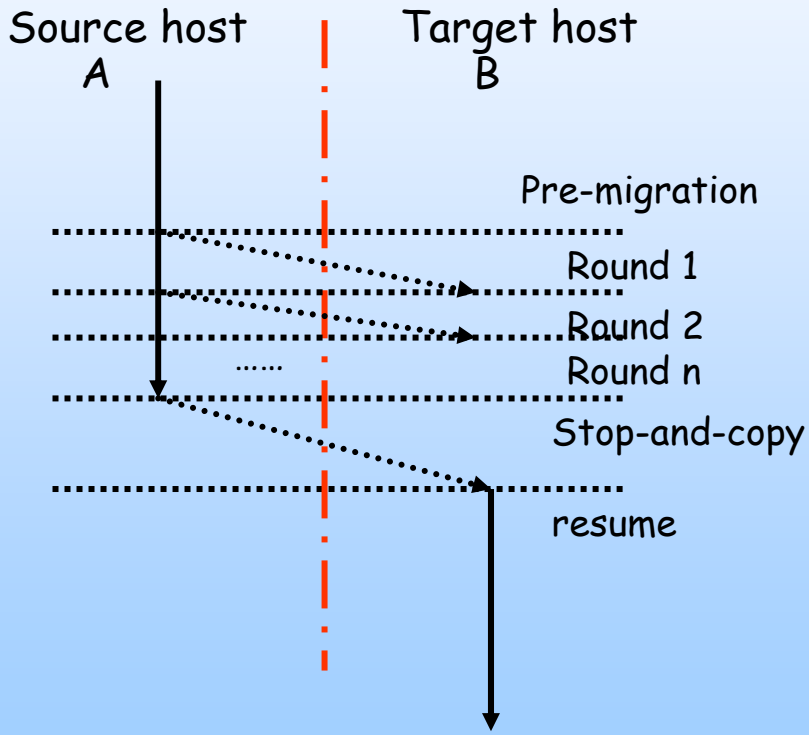
- It migrates whole OS with running applications rather than single processes in a nondisruptive fashion , avoiding residual dependencies.
- Applicable scenarios (cluster, data center)
  - Load Balancing
  - Online Maintenance and Fault Tolerance
  - Power Management for Green Computing

Live VM Migration

- VM migration issues
  - Memory
  - Network connection
  - Disk storage
- Performance metrics
  - Migration Downtime
  - Total Migration Time
  - Total Data Transferred
  - Migration overhead

Previous approaches

- Memory pre-copying: XenMotion, VMotion
  - Pre-Migration phase (select alternate physical host and resource reservation)
  - Memory push phase (pre-copying algorithm)
  - Stop-and-copy phase
  - Resume phase



Some shortcomings of XenMotion

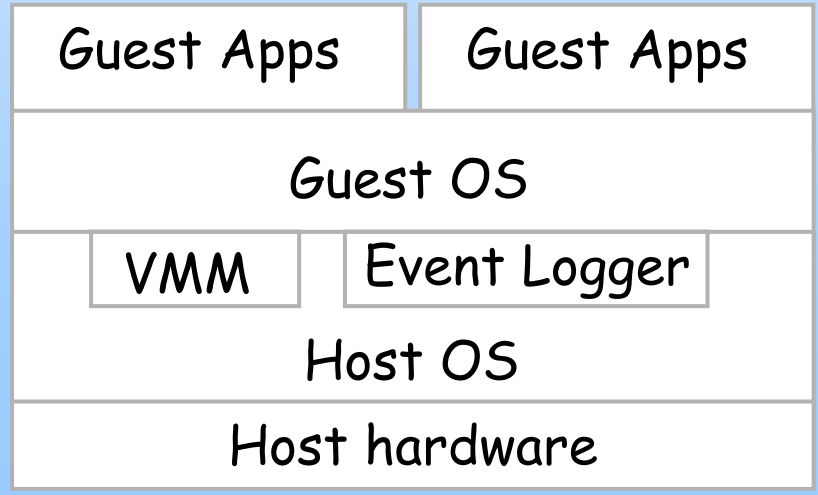
- The memory transferring speed should be much faster than the memory page dirtied;
- Memory-to-memory approach is only efficient in a LAN but bring little benefit for low-bandwidth WAN;
- Just copy memory, but can't copy the cache, TLB at the stop-and-copy phase;
- Some optimization may cause bad experience for services.

CR/TR-Motion: A novel VM migration approach

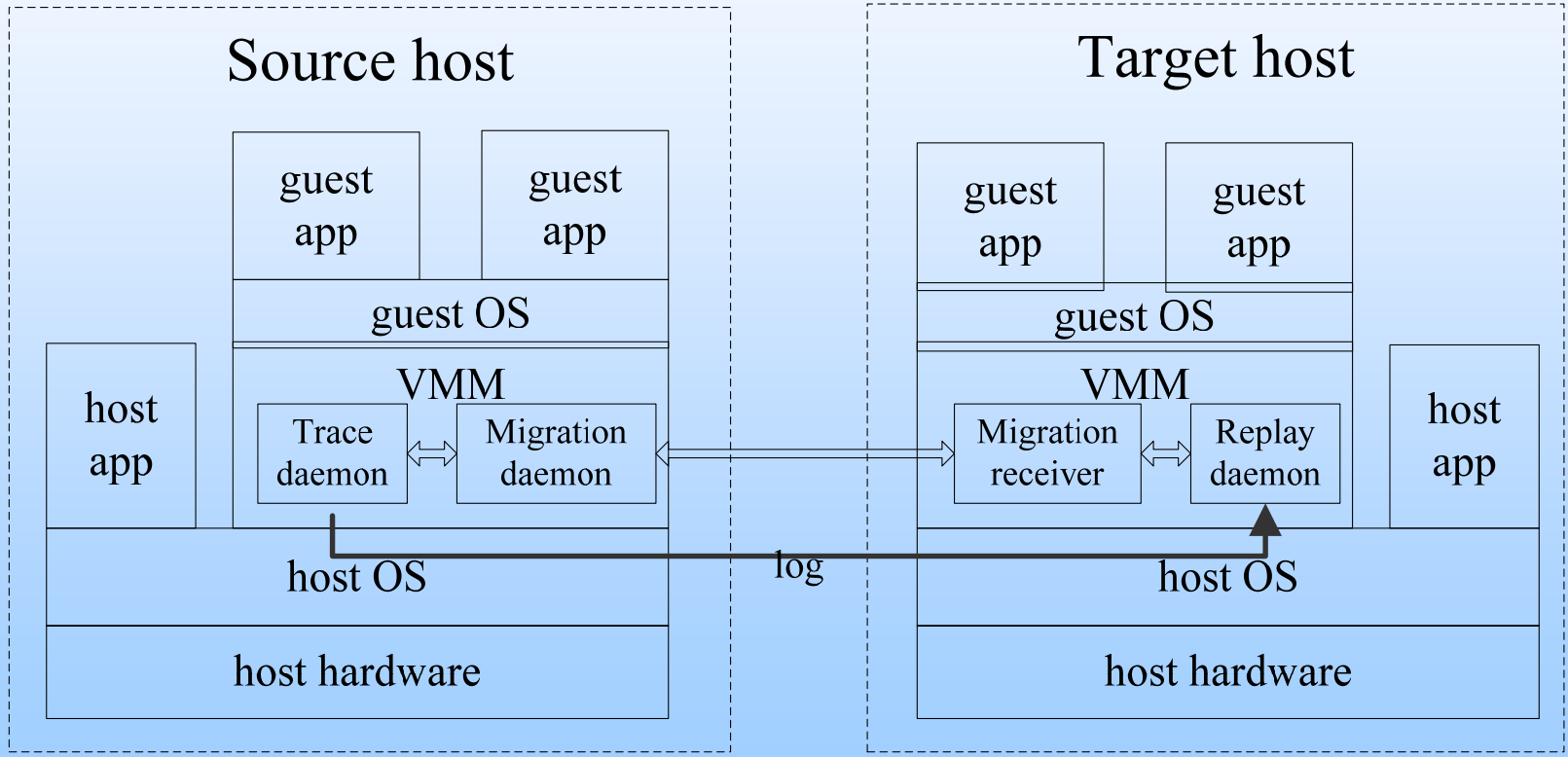
- ReVirt is adopted as our groundwork.
- Checkpointing/recovery combining with trace/replay technology are used to provide fast and transparent live VM migration.
- We orchestrate the running source and target VM with execution trace logged on the source host.

ReVirt overview

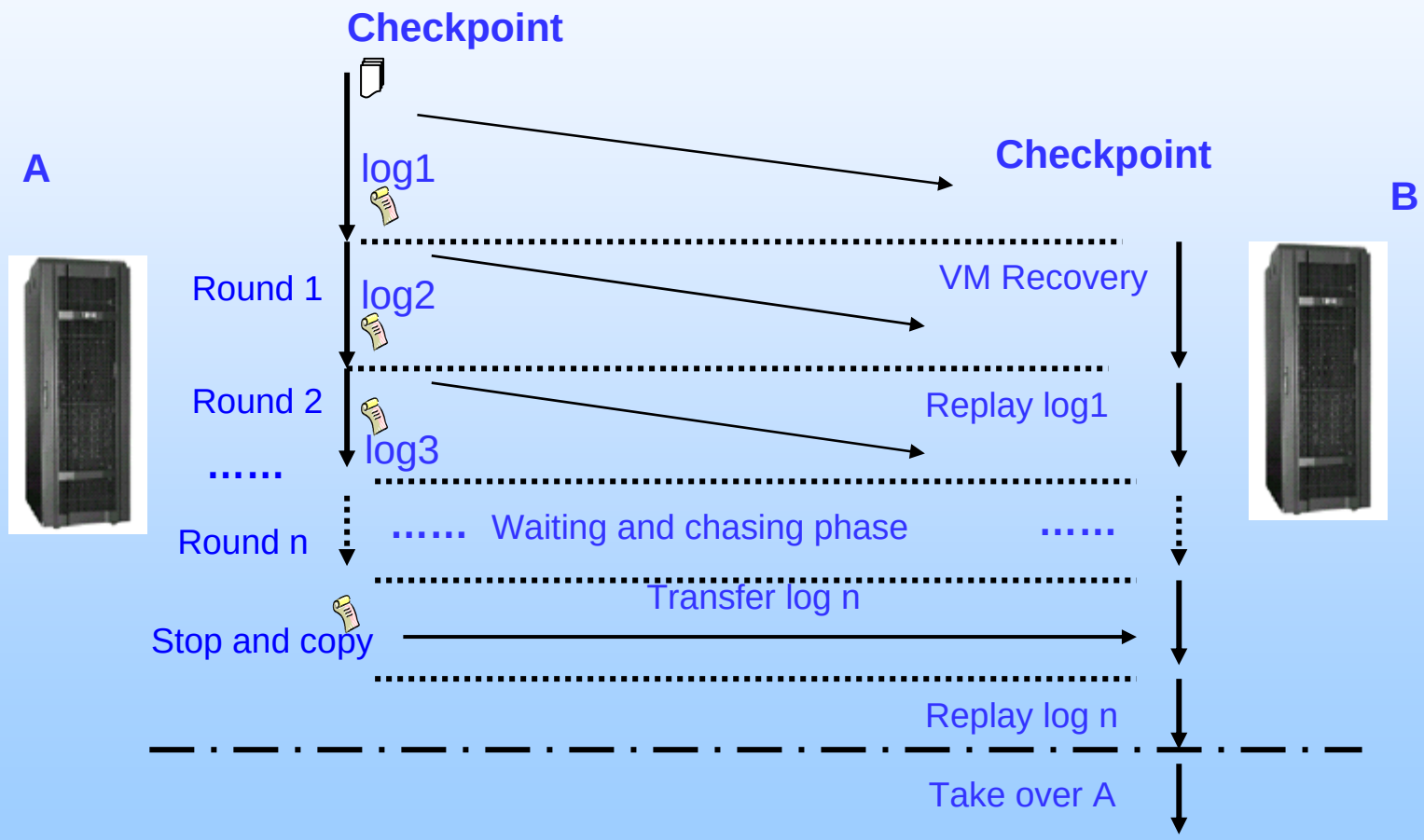
- ReVirt is a full-system trace and replay system, it was ported on UMLinux.
- ReVirt only log non-deterministic events (time and external input)
- ReVirt adds reasonable time and space overhead, Logging adds only 0-8% performance overhead, and log growth rates range from 40 MB per day to 1.2GB per day for different workloads .



CR/TR-Motion system structure



Migration process



Two prerequisite

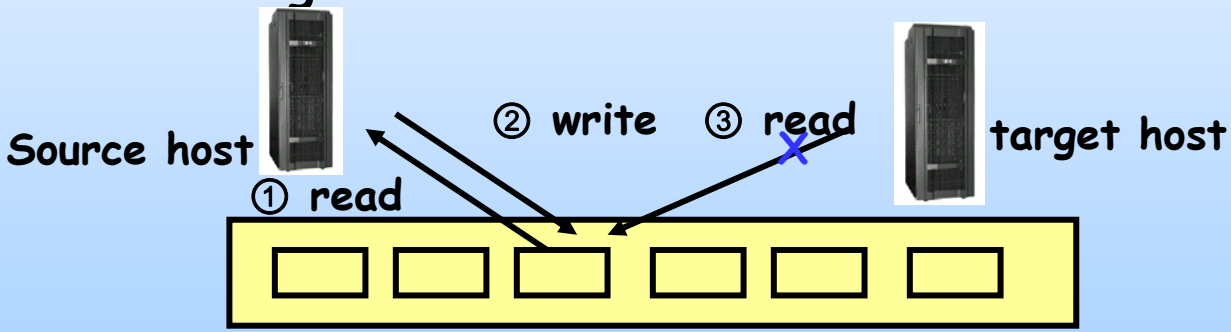
- Is log transfer speed faster than log growth speed?
- Is log replay speed on the target host faster than log growth speed?

log and replay rate in our experiments

Workloads	Log growth rate (KB/s)	Log replay rate (KB/s)	R_{replay}/R_{log}
daily use	7.9	290.7	36.8
kernel-build	1.2	1.27	1.05
static web app	247	402.6	1.63
dynamic web app	722	866.4	1.20
unixbench	61	65.88	1.08

Migration challenges

- Migrate block devices
  - NAS or SAN can be accessed uniformly from all host machines in the cluster.
  - Solve WAR (write after read) issues when replay on the target host in LAN.



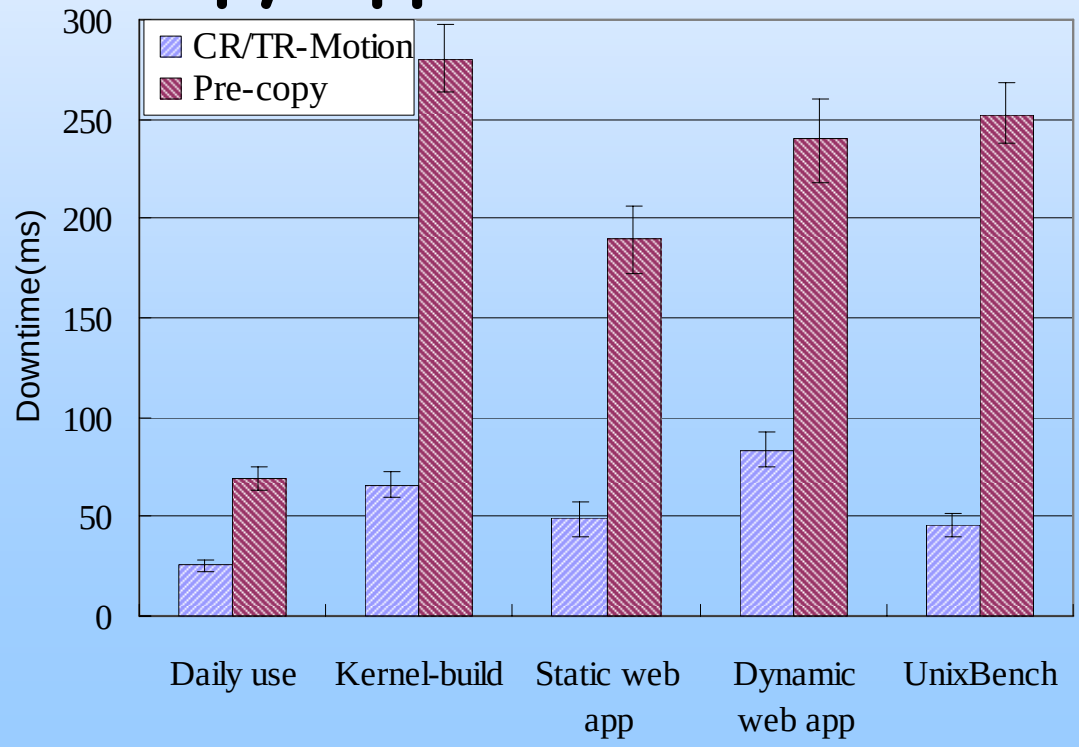
- Guarantee Network connections
  - The same solution as XenMotion (ARP reply in LAN)

Performance evaluation

- AMD Athlon 2500+ processor and 1GB DDR RAM, Intel Pro/1000 Gbit/s NIC (limit the network bandwidth to 500Mbit/sec)
- The virtual machine is configured to use 512MB of RAM.
- The VM being migrated is the only VM running on the source machine and there are no VMs running on the target machine.

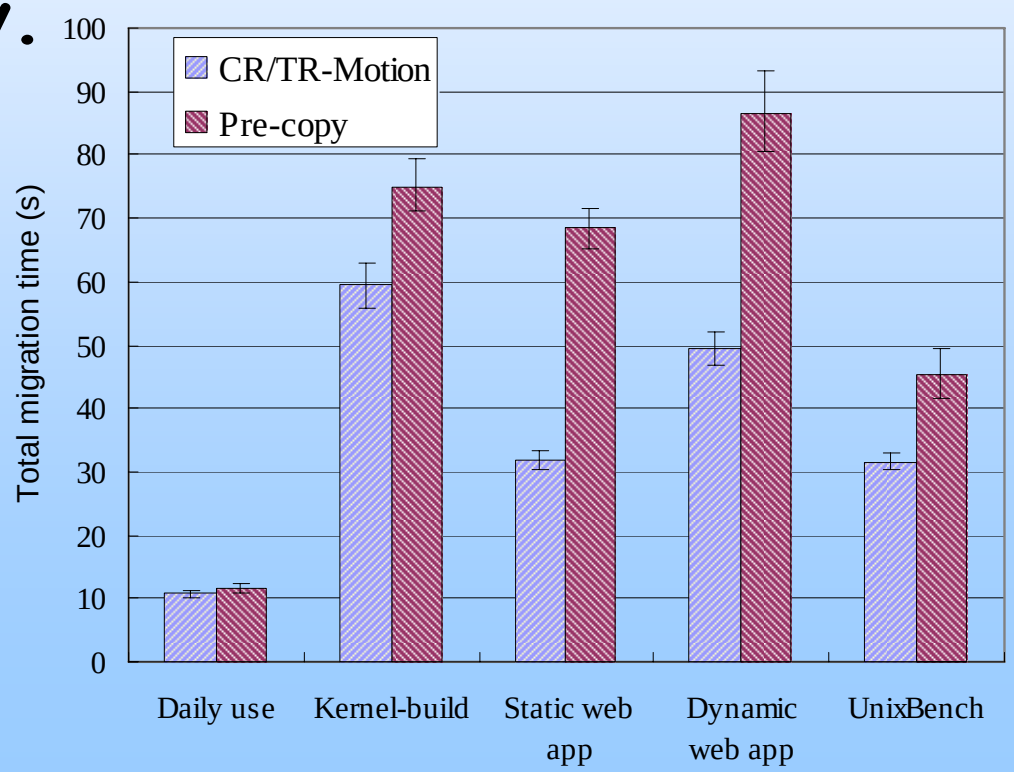
Migration downtime

- Our approach reduced migration downtime by 72.4% in average compared to pre-copy approach.



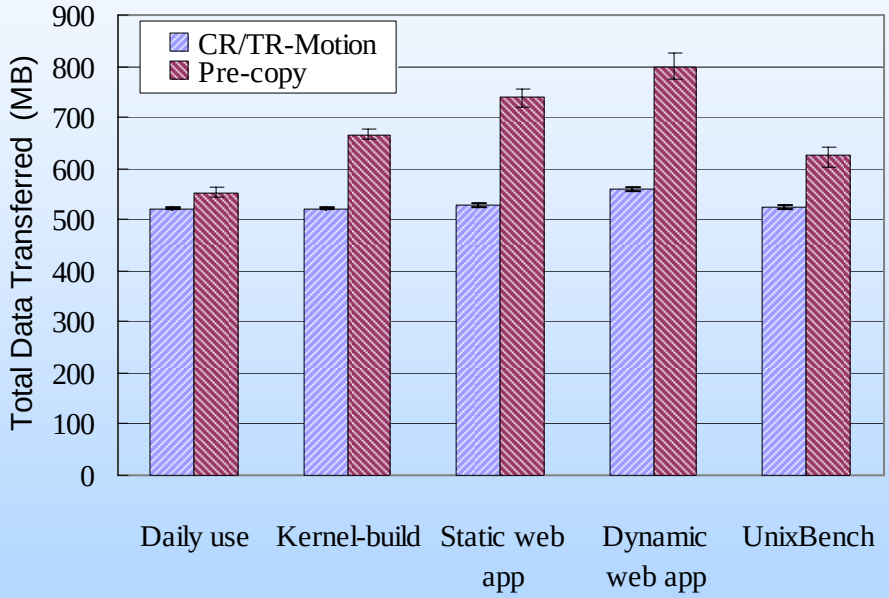
Total migration time

- Our approach reduces the total migration time by 31.5% in average compared to Pre-copy.



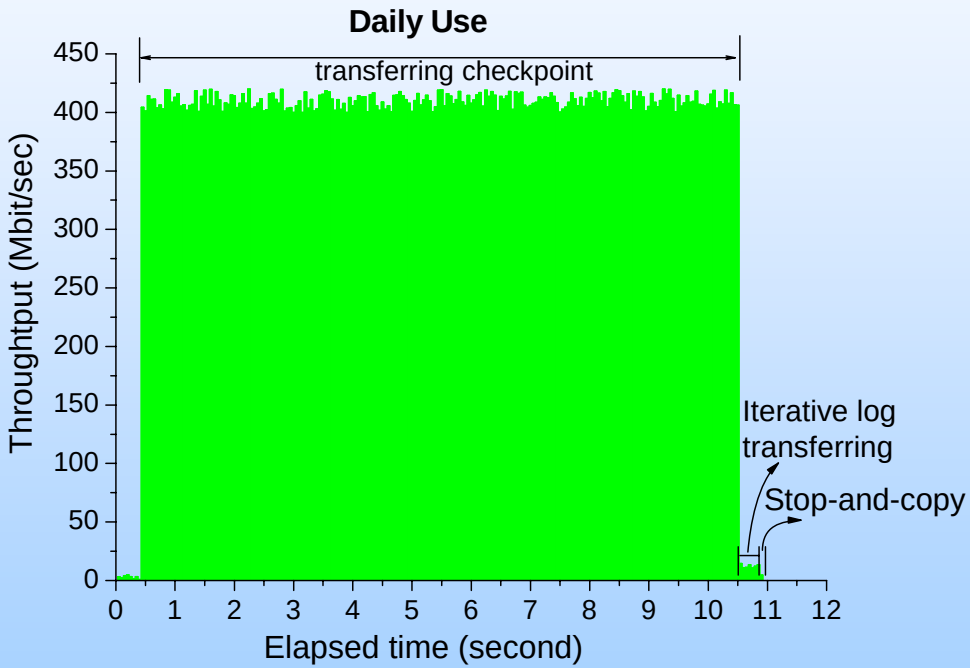
Total data transferred

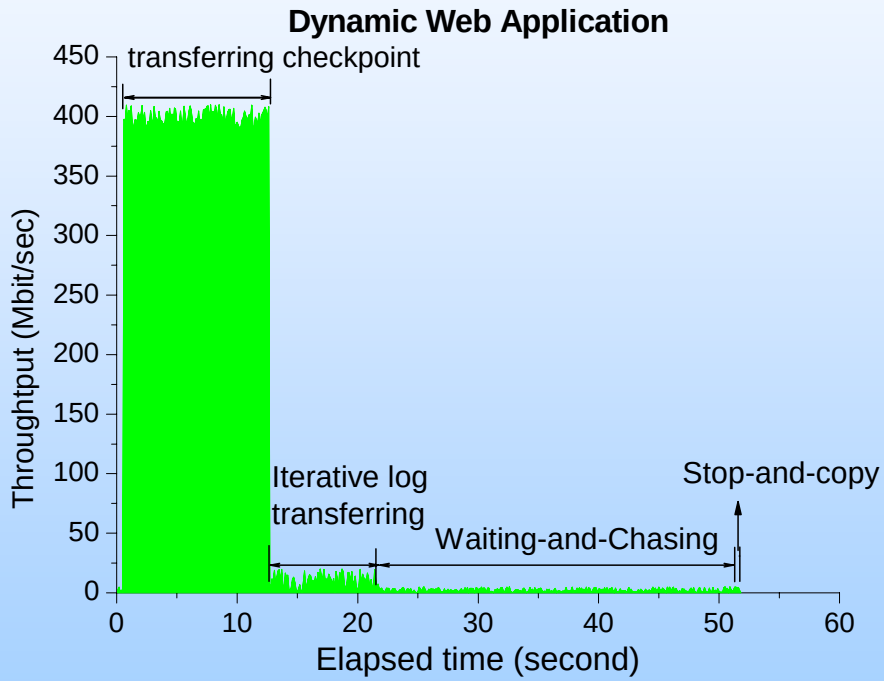
- CR/TR-Motion reduces synchronization traffic by 95.9% in average.
- This improvement will bring great benefit when our migration scheme is applied in low-bandwidth WANs.



Workloads	Synchronization data traffic (MB)		Reduction ratio
	CR/TR-Motion	Pre-copy	
daily use	0.48 (0.04)	38.54 (2.1)	98.8%
kernel-build	0.53 (0.06)	152.44 (8.2)	99.6%
static web app	8.34 (0.21)	228.99 (9.4)	96.4%
dynamic web app	36.4 (0.96)	288.05 (12.2)	87.4%
unixbench	2.59 (0.22)	113.38 (6.4)	97.7%

Network Throughput of Migration

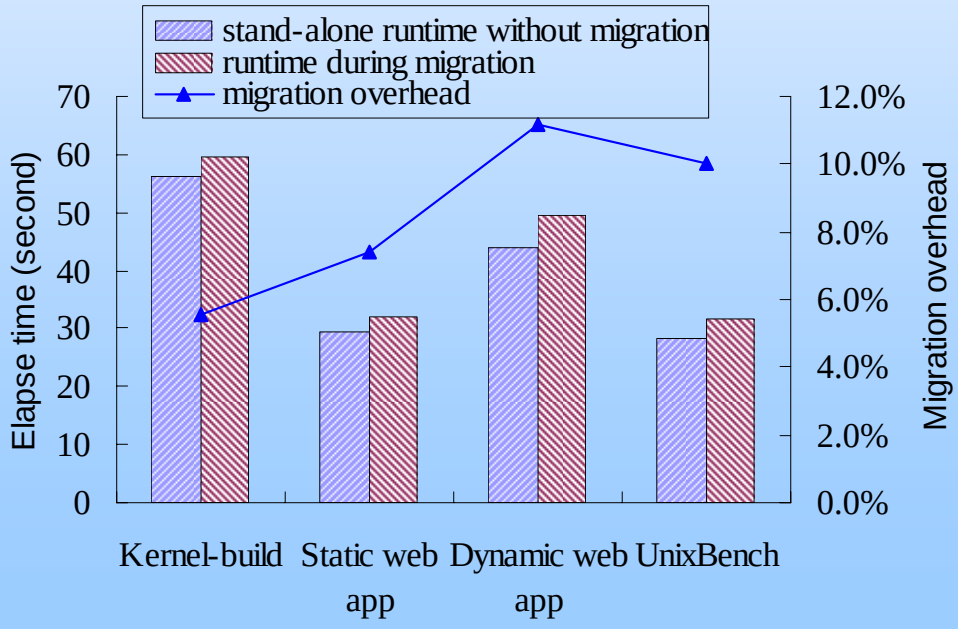


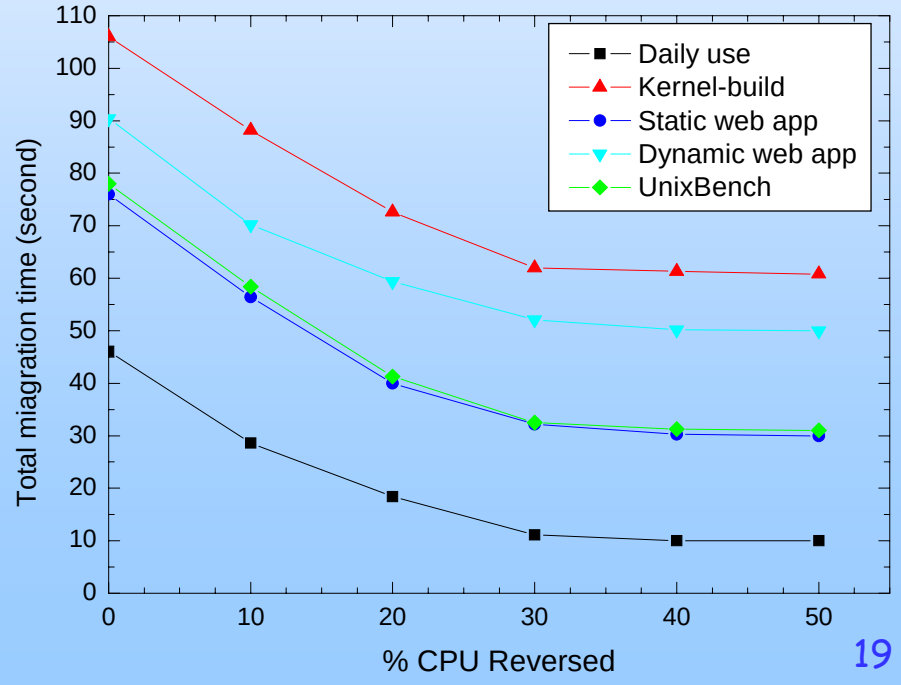


The above figures demonstrate that the migrations for those workloads cause reasonable network traffic and bandwidth consumption.

Migration overhead

- Overheads caused by transparent checkpointing and logging is less than 8.5% on average. It takes about 30% of a CPU to attain the maximum network throughput over the gigabit link.





Conclusion

- CR/TR-Motion provide a novel VM migration approach based on full-system trace and replay, that contrasts against memory-to-memory approach.
- Our scheme minimize the migration downtime and network bandwidth consumption.
- Our approach may bring a new benefit if the migration is performed in low-bandwidth WAN environments.

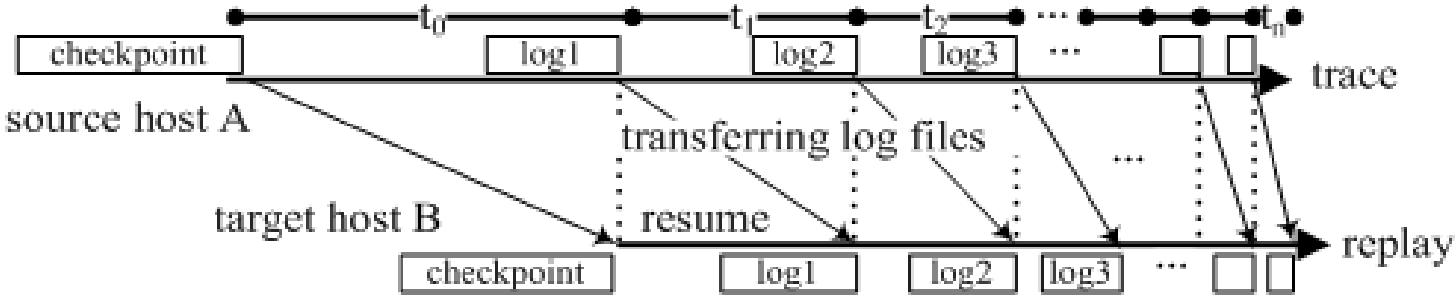
Questions and Comments!

Parameters definition

- Total Data Transmitted: TDT
- Transfer rounds: n
- Total Migration Time: TMT
- Time sequence to transfer instruction logs:(t_0,t_1,t_2,t_3,……,t_n)
- Log serial at each transfering rounds:(log_1,log_2,log_3,……,log_n)
- Checkpoint Volume: V_{ckpt}
- Log transfer Rate: R_{trans}
- Log growth Rate: R_{log}
- Log replay rate: R_{replay}
- V_{thd} = 1KB : the threshold value at which the iteration terminated
- φ: R_{log}/R_{trans},
- : R_{replay}/R_{log}

Scenario 1: fast synchronization

- $R_{replay} >> R_{log}$
- No need to execute the waiting-and-chasing phase
- The key factor is parameters: R_{trans} and R_{log}



Migration process of fast synchronization

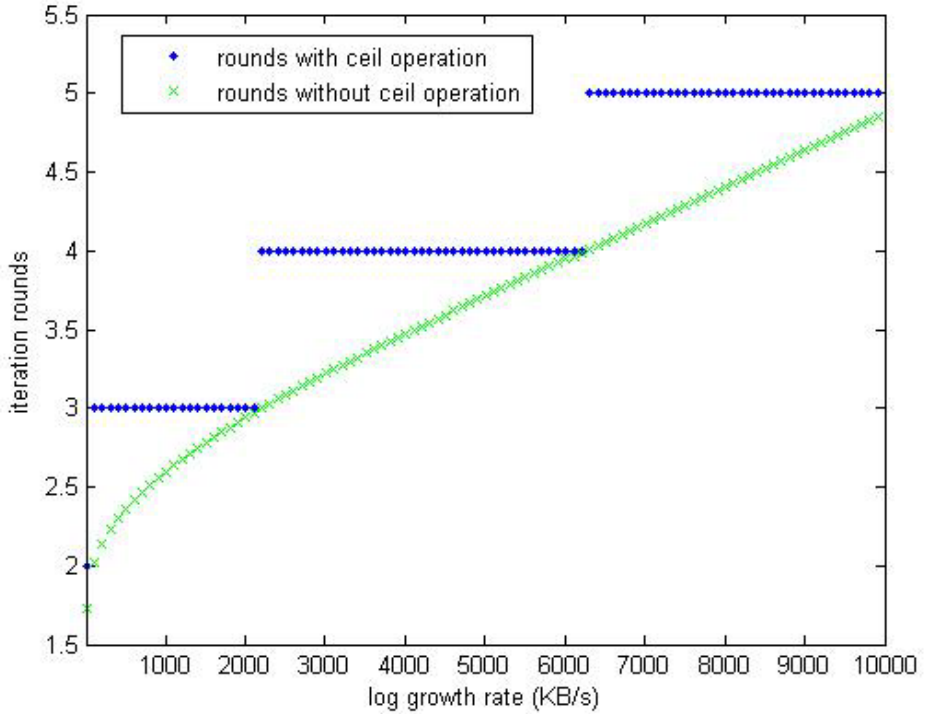
Iteration rounds (scenario 1)

- Supposing iteration terminated when the log size is less that V_{thd} =1kB

$$t_{(n-1)} R_{log} \le V_{thd}$$

i.e. $$V_{ckpt} \varphi^{(n-1)} \le V_{thd}$$

i.e. $$n = 1 + \left\lceil \log_{\varphi} \frac{V_{thd}}{V_{ckpt}} \right\rceil$$



R_{trans}=400Mbit/sec and V_{ckpt}=512MB

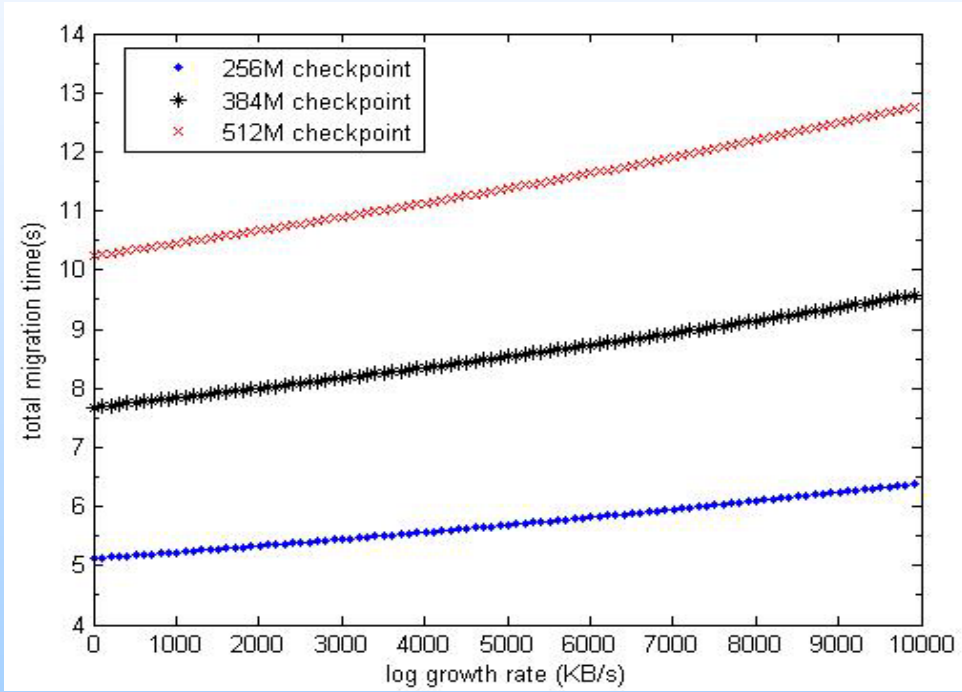
Total Migration Time (Scenario 1)

$$t_0 = \frac{V_{\text{checkpoint}}}{R_{trans}} \qquad t_1 = \frac{R_{\log} t_0}{R_{trans}} = \varphi t_0$$

.....

$$t_n = \frac{R_{\log} t_{(n-1)}}{R_{trans}} = \frac{V_{ckpt} R_{\log}^{(n-1)}}{R_{trans}^{n}} = t_0 \varphi^{(n-1)}$$

$$\text{TMT} = \sum_{i=0}^{n} t_i = \frac{V_{ckpt}(1-\varphi^n)}{R_{trans}(1-\varphi)}$$



R_{trans}=400Mbit/sec and V_{ckpt}=512MB

Conclusions (Scenario 1)

- **Total Data Transmitted**

$$\mathrm{TDT}=V_{ckpt}+\sum_{i=1}^{n}V_{log_i}=\mathrm{TMT}*R_{trans}=\frac{V_{ckpt}(1-\varphi^n)}{(1-\varphi)}$$

- **Downtime**

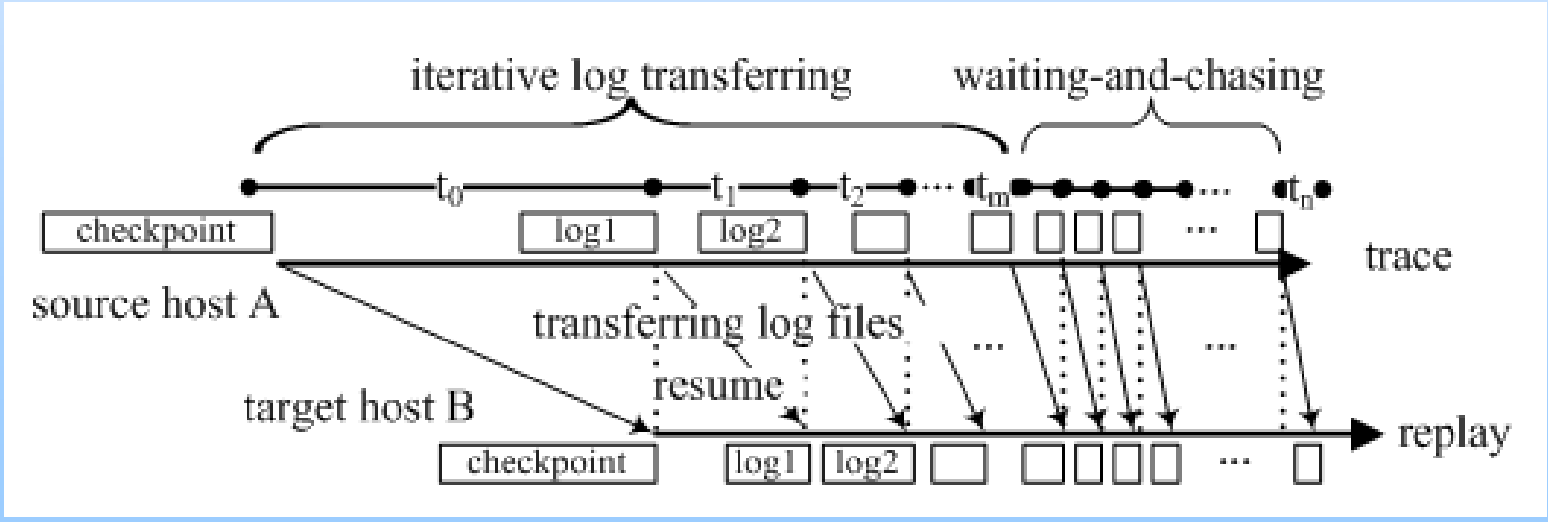
$$T_{downtime}=t_n+V_{log_n}/R_{replay}+t_{other}$$

Scenario 2: slow synchronization

- $R_{replay} > R_{log}$, but log file is not replayed fast enough.
- A waiting-and-chasing phase is performed to postpone the stop-and-copy phase.

Condition:

$$\frac{\sum_{i=1}^{m} V_{log_i}}{R_{log}} - \frac{\sum_{i=1}^{m} V_{log_i}}{R_{replay}} <= \frac{V_{ckpt}}{R_{trans}}$$



Migration process with a waiting-and-chasing synchronization phase

Conclusions (Scenario 2)

- Total Data Transmitted

$$TDT = V_{ckpt} + \sum_{i=1}^{n} V_{log_i} = (\frac{\partial \varphi}{\partial - 1} + 1) V_{ckpt}$$

- Total migration time

$$TMT = \frac{V_{ckpt} + V_{log_1}}{R_{trans}} + \frac{\sum_{i=1}^{n} V_{log_i}}{R_{replay}} = (\varphi + \frac{\partial}{\partial - 1}) \frac{V_{ckpt}}{R_{trans}}$$