ADAPTIVE LIVE VM MIGRATION
OVER A WAN
MODELING AND IMPLEMENTATION

Weida Zhang, King Tin Lam, Cho-Li Wang
Department of Computer Science, The University of Hong Kong
Live Migration of VMs

- Live migration: the VM is lively on the move
  - Dynamic resource provisioning within a data center
  - An enablement of cloud technology
  - Enhancing IT’s efficiency and cost-effectiveness.
Wide-area Live Migration (LM) !?

- **WAN App Scenarios:**
  - **Facilitate business operations:**
    - Recent report: Instagram migrated user photos from Amazon EC2 to Facebook VPC*
      * [How Facebook Moved 20 Billion Instagram Photos Without You Noticing](https://example.com)
  - **Mobile working env.**: a virtual workplace migrating from your home desktop PC to your smartphone, and then to your office workstation, and vice versa (OT!).
  - **Cloud federation**: move VMs from vendor to vendor
  - **Global job scheduling**: move the VM around the world
Overview

- Introduction
  - Related Work
  - Problem Description

- Methodology
  - Our Invention: A Fractional Hybrid-copy LM Framework
  - Methodology Overview
  - Profiling, Modeling & Simulation
  - Recursion
  - Implementation

- Experiments & Results
Introduction

Related Work

Problem Definition
Existing Work on Live Migration

- **Pre-copy** [2,3]
  - small downtime

- **Post-copy** [4]
  - zero downtime
  - performance penalty

[2] Nelson, USENIX’ 05
[3] Clark, NSDI’ 05
Pre-copy Algorithms

1. Green means synchronized; red means unsynchronized.
2. First iteration, all pages are copied.
3. Other iterations, only the pages dirtied in the previous iteration are copied.
4. If bandwidth > page dirty rate, there will be fewer and fewer dirty pages.
5. Freeze-and-copy phase, all unsynchronized pages are copied.
Post-copy Algorithms

(1) Resumes the VM in the destination immediately

(2) Background transferred pages turn green

(3) On-demand requested pages introduce performance penalties

Freeze and copy cpu states only
Zero service downtime

Performance degradation
Problem

- Problem: pure pre-copy and post-copy are not doing well on a WAN.
- Hybrid: tradeoff between downtime and performance penalty
Existing Work on Wide-Area LM

- Pre-copy memory & pre-copy storage [7,9]
  - [7] Akoush, MASCOTS’ 11
  - [9] Bradford, VEE’ 07
- Pre-copy memory & post-copy storage [11,13]
  - [13] Luo, CLUSTER’ 08
- Pre-copy memory & hybrid-copy storage [14] = Pre-copy memory & pre-copy S% of storage
  - [14] Zheng, VEE’ 11

Our contribution of a new approach:

- A fractional hybrid-copy = Pre-copy M% memory & pre-copy S% storage
- Adaptive = Fractional + Model to find (M, S)
Methodology

A Fractional Hybrid-copy LM Framework
Methodology Overview: An Adaptive Process
Profiling, Modeling and Simulation
Recursive Searching of $(M, S)$
Implementation
Fractional Hybrid-copy

![Diagram showing pre-copy, freeze-and-copy, and post-copy phases with downtime and storage/memory labels.](image-url)
Fractional Hybrid-copy

(1) At time 0, nothing is copied yet

(2) S storage are migrated initially

(3) Tried the best to migrate M% memory iteratively

(4) M and S are migrated before post-copy

(5) 100% memory are migrated during post-copy

(6) 100% storage are migrated during post-copy
Methodology Overview

1. Network Condition & App behavior
2. Profiling
3. Predict the performance suppose M & S is given
4. Apply the policy regarding the prediction (whether/how to go on)
5. Do the migration / cancel the request
Methodology Overview

- **Profiling**: Network Condition & App behavior
- **Modeling**: Predict the performance, suppose \(\{\lambda\}, M, S\) is given
- **Simulation**: Have to be fast
- **Implementation of fractional hybrid-copy**

**Steps**:
1. **Profiling**
   - Network Condition & App behavior
   - \(\{\lambda\}\)

2. **Modeling**
   - Predict the performance
   - Suppose \(\{\lambda\}, M, S\) is given
   - \(\text{dirty}(\tau)\)

3. **Simulation**
   - Have to be fast
   - Policy regarding the prediction (whether/how to go on)
   - \(\{\lambda\}, M, S \& \text{dirty}(\tau)\)
   - \(\Gamma\)

4. **Implementation**
   - Do the migration / cancel the request
   - Recursion (ternary search)

**Equation**:
- \[\lambda, M, S \Rightarrow \text{dirty}(\tau)\]
Profiling, Modeling & Simulation

- Key components of simulation: dirtying rate
  - Constant dirtying rate [10]
    - Simple profiling: count how many pages are updated
    - O(1) simulation
  - Full-history profile + replay-based dirtying rate [10]
    - Heavy profiling overhead: record every update of a page
    - O(N), N is the size of memory or storage
  - Assuming Poisson distribution
    - Reduced overhead: how many times a page is updated
    - n samples, one λ for each page/trunk
    - O(n)

[10] Akoush, MASCOTS’ 10
Performance restoration agility, $\Gamma \leftarrow$ Our proposed new metric

- $\Gamma$ is the variable to be optimized
- $\Gamma$ is a function of profile $\{\lambda\}, M, S, D$
- $\Gamma = \frac{\delta T}{(D + \Delta T)}$
  - $\delta T$: a configurable time, we use 20 seconds
  - $\Delta T$: time needed for the VM at restore to execute the workload of $\delta T$ during normal execution
- $\Gamma = \frac{1}{(D \times weight_1 + \text{Penalty} \times weight_2)}$
  - you can use different policies to balance downtime and penalty, i.e. balance between pre-copy and post-copy
1-dimensional View of $\Gamma$

- 0 downtime pre-copy possible
- 0 downtime pre-copy impossible

Graphs showing the relationship between $\Gamma$ and $M$ with a vertical line indicating the transition point.
Recursion: Searching for M & S

1. Assume the M is magically instant-copied (greedy)
   - Find S using Ternary Search
   - Assume the S could be live pre-copied, i.e. 0-down time pre-copy possible
   - If the migration of storage-only cannot be live, there is no way to do the live migration

2. Fix the found S
   - Find M using Ternary Search
Ternary Search (Magical $M$, sysbench)
Ternary Search (Fixed S, v8)
Implementation

- Implemented on Xen
Experiments & Results
Experimental Settings

- v8 benchmark (JavaScripts on Google v8 engine)
- Sysbench (intensive read/write operations)
- Move VM from A to B, migration channel separated from application’s network channel
- Migration channel: 5ms RTT, 40 Mbps (two ends within a city)
Result 1: Predictabilities (Memory, v8)

TABLE I. OVERALL EVALUATION OF THE MEMORY PREDICTION

<table>
<thead>
<tr>
<th></th>
<th>Read</th>
<th></th>
<th>Write</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$j^*$</td>
<td>$j_{actual}$</td>
<td>$j^*$</td>
<td>$j_{actual}$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>59.8%</td>
<td>0</td>
<td>54.1%</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>10.1%</td>
<td>1</td>
<td>3.5%</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3.8%</td>
<td>0</td>
<td>54.1%</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>26.3%</td>
<td>1</td>
<td>0.4%</td>
</tr>
<tr>
<td>accuracy$_R$</td>
<td>86.1%</td>
<td></td>
<td>accuracy$_W$</td>
<td>96.1%</td>
</tr>
</tbody>
</table>
**Result 1: Predictabilities (Storage, sysbench)**

<table>
<thead>
<tr>
<th></th>
<th>Read</th>
<th></th>
<th>Write</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$j^*$</td>
<td>$j_{\text{actual}}$</td>
<td>$j^*$</td>
<td>$j_{\text{actual}}$</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>75.1%</td>
<td>0.9%</td>
<td>0</td>
<td>96.6%</td>
</tr>
<tr>
<td></td>
<td>2.2%</td>
<td>21.9%</td>
<td>1</td>
<td>0.0%</td>
</tr>
<tr>
<td>accuracy$_R$</td>
<td>96.9%</td>
<td></td>
<td>accuracy$_W$</td>
<td>96.6%</td>
</tr>
</tbody>
</table>
Result 1: Predictabilities (Simulation, v8)

- When $(M,S) = (60\%, 50\%)$
- $\Gamma = 20\%$

<table>
<thead>
<tr>
<th>TABLE III.</th>
<th>PREDICTION OF $T$, $U$ AND $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Predicted (s)</td>
</tr>
<tr>
<td>Total migration time ($T$)</td>
<td>1063.3</td>
</tr>
<tr>
<td>Remote uptime ($U$)</td>
<td>554.3</td>
</tr>
<tr>
<td>Downtime ($D$)</td>
<td>49.2</td>
</tr>
</tbody>
</table>
Result 2: Search of \((M, S)\) (v8)

Searching of \(S\) when \(M\) is magically copied

Searching of \(M\) when \(S = 3\)
Found \(M = 95\%\) is the best

Whole-page overwriting technique:
We found if a whole page-writing (4K) causes a fault during post-copying, it is good to just overwrite the page, without remote fetching the page.
Result 3: Overall Performance

v8 (M, S) = (48%, 0%)

Sysbench = (98%, 25%)
Conclusions

- Generalized the hybrid combination of memory and storage migration by \((M, S)\)
- Defined the restoration agility, Gamma, to describe the liveliness/performance of a \((M, S)\) migration
- Proposed a method to find the best \((M, S)\) pair to achieve good restoration agility
  - Improved prediction with profiling and dirtying rate function
  - Ternary search of \((M, S)\)
- Unique implementation of fractional hybrid copy
Thanks and Q&A!
Pre-copying of Storage

- Pre-copy using queue-based data structure
- Iterative precopy described in Clark et al.
- Batched iterative precopy implemented in Xen
Post-copying of Storage

Upon writes to a block, requeue the block

The blocks are migrated in the order from the least volatile to the most volatile
Post-copy of Memory (Miss)

PAGE_POSTCOPY

EIP: read $virtual_address

try again

Y

is request sent?

N

tell xc_restore to send request

page fault!

miss
Post-copying of Memory (Hit)

EIP: read $virtual_address

try again

if _PAGE_POSTCOPY, page fault!
but the data is already arrived
remove the bit