Traffic-Sensitive Live Migration of Virtual Machines

Umesh Deshpande
Computer Science, Binghamton University

Dr. Kate Keahey
Mathematics and Computer Science Division, Argonne National Lab

Contact: umesh.deshpande@binghamton.edu
Background: Live VM Migration

- Migration of a running Virtual Machine between hosts
- Transfer

- **Pre-copy** live VM migration
  - 1\textsuperscript{st} Iteration
    - Transfers entire memory
  - 2\textsuperscript{nd} Iteration
    - Dirty Pages

- **Post-copy** live VM Migration
  - Preparation (live)
  - Resume Time (live)
  - Active Push + Demand Paging
Motivation: Migration of VMs

- Shutting down rack for cluster maintenance
- Imminent failures
- Power Saving
Problem

• Migration of Network-bound VMs
  – Transfer of Gigabytes of memory
  – Contention between VM application and migration traffic at the NICs

• Contention depends upon direction of traffic
  – Flows in the same direction compete
  – Flows in opposite direction complement
Problem

• Migration traffic competes with
  – **Pre-copy:** Outbound VM application traffic at source
  – **Post-copy:** Inbound VM application traffic at destination

![Diagram of VM traffic and migration traffic]

• Effect of contention
  – **Prolongs Migration**
  – **Degrades VM applications**
Problem

- Contention during migration depends upon
  - VM’s predominant traffic direction
  - VM migration technique selected

- Effect of contention
  - Prolongs Migration
  - Degrades VM applications
Solution: Traffic-sensitive migration

• **Goal:** Reduce contention at migration end-points for migration of co-located VMs

• Select migration technique for each VM
  – Direction of most VM traffic complements the direction of migration traffic
Existing Solutions

• Post-copy: Transfers each page only once
• Content optimization:
  – Shrinker, Gang Migration, VMFlock
  – Compression, Differential compression, Deduplication

• Migration of Virtual Clusters
  – VCT: Non-live migration of VMs and disk images
  – VC Migration: Compares different strategies for migration of multiple VMs
1. Periodically measure TX and RX traffic rate for each VM

2. Calculate severity possible contention with every combination of pre-copy and post-copy
   - E.g. (vm1, vm2, vm3) : (pre, pre, post), (pre, post, pre)...

3. Select the one that yields the least contention
For each combination

• Source contention = \( \Sigma \) Rate of outgoing traffic for VMs migrated with pre-copy + Outgoing background traffic

• Destination contention = \( \Sigma \) Rate of incoming traffic for VMs migrated with post-copy + Incoming background traffic

• Contending Traffic = \textbf{Max} (Source contention, Dest. Contention)
Two co-located VMs | Tx Rate  | Rx Rate  
---|---|---
VM1 | 200 Mbps | 400 Mbps
VM2 | 300 Mbps | 500 Mbps

1. VM1 pre-copy, VM2 pre-copy
   - Source contention = 500 Mbps
   - Destination contention = 0
   - Contention = Max (500, 0) = 500 Mbps

2. VM1 post-copy, VM2 pre-copy
   - Source contention = 300 Mbps
   - Destination contention = 400 Mbps
   - Contention = Max (300, 400) = 400 Mbps
Implementation: Networking

- Implemented on KVM/QEMU platform
- 1Gbps Ethernet interconnect

Virtual Networking in KVM/QEMU
Implementation

init_migration(technique);

Traffic monitoring information

Central Server

init_migration(technique);

Traffic monitoring information

VM1

................

MM

QEMU

Gather traffic information periodically

Iptables

Source Host1

VMn

................

MM

QEMU

Gather traffic information periodically

Iptables

Source Hostn

VM1

................

MM

QEMU

Gather traffic information periodically

Iptables

Source Hostn
Evaluation

• Compare Against: Pre-copy only, Post-copy only
• Configuration
  • Host: 8 CPUs, 16GB memory, VM: 2 vCPUs, 5GB memory
  • VM1: Netperf client, VM2: Netperf server (VM1 → VM2)
Evaluation

- Compare Against: Pre-copy only, Post-copy only
- Configuration
  - Host: 8 CPUs, 16GB memory, VM: 2 vCPUs, 5GB memory
  - VM1: Netperf client, VM2: Netperf server (VM1 → VM2)
Evaluation

- Compare Against: Pre-copy only, Post-copy only
- Configuration
  - Host: 8 CPUs, 16GB memory, VM: 2 vCPUs, 5GB memory
  - VM1: Netperf client, VM2: Netperf server (VM1 → VM2)

<table>
<thead>
<tr>
<th></th>
<th>Pre-copy</th>
<th>Post-copy</th>
<th>Traffic-sensitive Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Migration Time</td>
<td>79.1</td>
<td>92.1</td>
<td>48.2</td>
</tr>
<tr>
<td>(seconds)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amount of Data Transferred (MB)</td>
<td>10280</td>
<td>10277</td>
<td>10278</td>
</tr>
<tr>
<td>Netperf Performance (Mbps)</td>
<td>690.47</td>
<td>660.05</td>
<td>894.65</td>
</tr>
</tbody>
</table>

- TMT: 42% and 49% lower than pre-copy and post-copy
- Performance: 29% and 35% higher than pre-copy and post-copy
Evaluation

- 8 Source Hosts, each host runs 2 VM
- 12 VMs run Redis database server
- 4 VMs query with YCSB workload
  - Insert, read, update queries

<table>
<thead>
<tr>
<th></th>
<th>Without Migration</th>
<th>Pre-copy</th>
<th>Post-copy</th>
<th>Traffic-sensitive Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Migration Time (seconds)</td>
<td>-</td>
<td>50.56</td>
<td>60.48</td>
<td>37.79</td>
</tr>
<tr>
<td>Total Migration Time (seconds)</td>
<td>-</td>
<td>74.5</td>
<td>139</td>
<td>57.75</td>
</tr>
<tr>
<td>Amount of Data Transferred (GB)</td>
<td>-</td>
<td>50.90</td>
<td>30.18</td>
<td>34.07</td>
</tr>
<tr>
<td>YCSB Performance (Operations / second)</td>
<td>4802</td>
<td>3875</td>
<td>4161</td>
<td>4126</td>
</tr>
</tbody>
</table>

- TMT reduction: 23% vs pre-copy, 59% vs post-copy
- Vs. Pre-copy: 6% lesser degradation, 68% lower network traffic overhead
Future Work

• Migration from same source host to different destination hosts
  • Scattering or consolidation of VMs
  • Considering the combinations across the hosts

• Account for the traffic at the destination host to selecting a suitable destination
Conclusions

• Combination of pre-copy and post-copy to reduce network contention
  • Esp. for VMs with unidirectional traffic

• Reduces total migration time
  • Allows faster eviction

• Minimizes application network-bound degradation
Thanks!

Contact: umesh.deshpande@binghamton.edu

Website: http://www.umeshhome.com