LIVE MIGRATION

組員： 7099056034  陳俞孝
       7099056037  林正浩


OUTLINES

* INTRODUCTION

* PAPER STUDYING
  * Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud
  * Albatross: lightweight elasticity in shared storage databases for the cloud using live data migration
  * Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments

* CONCLUSION
INTRODUCTION
Live migration allows a server administrator to move a running virtual machine or application between different physical machines *without disconnecting the client or application.*

For a successful live migration, the memory, storage, and network connectivity of the virtual machine needs to be migrated to the destination.
Compared with traditional suspend/resume migration, live migration holds many benefits such as

- Energy saving.
- Load balancing.
- Online maintenance.
Why Live Migration? And Implementations!

* **Energy Saving**

  * Power savings can be made during periods of lower processing demand by migrating VMs from a host and powering it off until the extra capacity is required again.
Load Balancing

* VMs can be migrated to a specific host at any time in order to optimize VM performance, cluster balancing or consolidation ratios, without negatively impacting users.
Online Maintenance

A host requiring maintenance can have the VMs migrated from it, the work performed and be put back into service during normal business hours, without interrupting service.
Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud
Yi Zhao and Wenlong Huang

Institute of Computing Technology, Chinese Academy of Sciences Beijing 100190, China
SYSTEM ARCHITECTURE

Figure 1 System Architecture
Figure 2 virtual machine migration
* Load Balancing Algorithm

**Algorithm COMPARE_AND_BALANCE**

```plaintext
for each VM i on current host cur do
    for all host j = [m] do
        \( n_j \leftarrow \text{num of VMs on host } j \)
    end for
    \( n \leftarrow \sum_{j=1}^{m} n_j \)
    choose \( k \in [m] \) randomly with \( P[k] = \frac{n_k}{n} \)
    \( c' \leftarrow \text{MEASURE_COST}(k) \)
    \( c \leftarrow \text{MEASURE_COST}(\text{cur}) \)
    if \( c' < c \) then
        with probability \( c - c' \) migrate to \( k \)
    end if
end for
```

\[ c_i = \alpha \times \text{(cpu usage)} + \beta \times \text{(IO usage)} \quad (\alpha, \beta \in [0 \sim 1], \alpha + \beta = 1) \]
EXPERIMENT 1 : *Live migration time of virtual machine*
Three Tests:

1. Under the condition of having shared storage, testing the live migration command of **OpenVZ**:
   * “vzmigrate --online <host> VEID”

2. Under the condition of having no shared storage, testing the live migration command of **OpenVZ**:
   * “vzmigrate --online <host> VEID”
Three Tests (cont.):

3. Under the condition of having shared storage, testing the composite of the two commands:
   * “vzctl chkpt VEID --dumpfile <path>”
     * The command saves all the state of a running VE to the dump file on the shared storage and stops the VE
   
   * “vzctl restore VEID --dumpfile <path>”
     * The command restores the VE on the target host node via the dumped file on the shared storage generated by the previous command.
Environment:

Template of VM:
- fedora-core-5-i386-default.tar.gz
- 301.26MB

OpenVZ
- 2.6.1892.1.18.el5.028stabilo60.2
EVALUATION

**TABLE I.** LIVE MIGRATION TIME OF VE BETWEEN TWO HOSTS UNDER DIFFERENT CONDITIONS

<table>
<thead>
<tr>
<th>TEST</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>141.8</td>
<td>54.16</td>
<td>4.55</td>
<td></td>
</tr>
<tr>
<td>149.1</td>
<td>53.89</td>
<td>2.60</td>
<td></td>
</tr>
<tr>
<td>161.8</td>
<td>52.02</td>
<td>3.23</td>
<td></td>
</tr>
<tr>
<td>174.0</td>
<td>53.47</td>
<td>3.43</td>
<td></td>
</tr>
<tr>
<td>163.9</td>
<td>55.73</td>
<td>2.83</td>
<td></td>
</tr>
<tr>
<td>161.9</td>
<td>51.98</td>
<td>4.12</td>
<td></td>
</tr>
<tr>
<td>162.0</td>
<td>54.28</td>
<td>2.93</td>
<td></td>
</tr>
<tr>
<td>178.2</td>
<td>52.44</td>
<td>3.30</td>
<td></td>
</tr>
<tr>
<td>161.3</td>
<td>51.86</td>
<td>3.46</td>
<td></td>
</tr>
<tr>
<td>168.7</td>
<td>52.22</td>
<td>2.57</td>
<td></td>
</tr>
<tr>
<td>AVERAGE</td>
<td>162.3</td>
<td>53.21</td>
<td>3.30</td>
</tr>
</tbody>
</table>
EVALUATION

EXPERIMENT 2: Convergence
Environment:

Node144 functions as the shared storage and there are not virtual machines running on it.
TABLE IV. DISTRIBUTION OF VM ON HOSTS IN THE SECOND TEST

<table>
<thead>
<tr>
<th>Host</th>
<th>Node131</th>
<th>Node132</th>
<th>Node133</th>
<th>Node235</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num of VE</td>
<td>10</td>
<td>13</td>
<td>8</td>
<td>37</td>
</tr>
</tbody>
</table>

Figure 4 Standard deviation according to the distribution in table 4
Albatross: lightweight elasticity in shared storage databases for the cloud using live data migration
AUTHORs

* Sudipto Das, Divyakant Agrawal, and Amr El Abbadi
  * University of California

* Shoji Nishimura
  * NEC Corporation
When deployed over a pay-per-use infrastructure, elastic scaling and load balancing, enabled by low cost live migration of tenant databases, is critical to tolerate load variations while minimizing operating cost.

Albatross migrates the database cache and the state of active transactions to ensure minimal impact on transaction execution while allowing transactions active during migration to continue execution.
To be effectively used for elasticity, database migration must be lightweight.
  * i.e. with minimal service interruption and negligible performance impact.

Goal:
  * Design an efficient and low cost technique for live migration of a tenant database in a multitenant DBMS.
* **Multitenancy**
  * Resource sharing amongst different tenants.
  * Important to serve applications that have small but varying resource requirements.

* **Tenant**
  * An application’s database instance with its own set of clients and data.

* **Tenant Cell (or Cell)**
  * Represent a self-contained granule of application data, meta data, and state representing a tenant.

* **Shared process model**
  * The tenants share the database process.
A TM (transaction manager) consists of a concurrency control component for transaction execution and a recovery component to deal with failures.
**Service unavailability**
- Duration of time for which a tenant is unavailable during migration.

**No. of failed requests**
- Include both aborted transactions and failed operations.

**Impact on response time**
- Change in transaction latency (or response time) observed as a result of migration.

**Data transfer overhead**
- Data transferred during migration.
Most traditional DBMSs do not support live migration
- Since enterprise infrastructures were statically provisioned for peak expected load, and elasticity was not considered a first class feature in database systems.

Stop and migrate techniques
- A cell is migrated by
  - Stop the cell at the source DBMS node
  - Abort all active transactions
  - Flushing all the changes to the NAS
  - Restarting it at the destination.
- High migration cost
Albatross aims to have minimal impact on tenant SLAs while leveraging the semantics of the database structures for efficient database migration.

Achieved by iteratively transferring the database cache and the state of active transactions.
Figure 2: Migration timeline (times not drawn to scale).
**Phase 1: Begin Migration:**

- Migration is initiated by creating a snapshot of the database cache at $N_{src}$.
- This snapshot is then copied to $N_{dst}$.
- $N_{src}$ continues processing transactions while this copying is in progress.
**Phase 2: Iterative Copying:**

- At every iteration, $N_{dst}$ tries to “catch-up” and synchronize the state of $C_{migr}$ at $N_{src}$ and $N_{dst}$.

- $N_{src}$ tracks changes made to the database cache between two consecutive iterations.

- In iteration $i$, changes made to $C_{migr}$’s cache since the snapshot of iteration $i-1$ are copied to $N_{dst}$.

- This iterative phase continues until the amount of data transferred in successive iterations is approximately the same, i.e. $\Delta i \approx \Delta i-1$.
  - When $\Delta i \approx \Delta i-1$, irrespective of the magnitude of $\Delta i$, little gain is expected from subsequent iterations.
THE Albatross TECHNIQUEs

* Phase 3: Atomic Handover:

1. Flush changes from all committed transactions at $N_{src}$.

2. Synchronize the remaining state of $C_{migr}$ between $N_{src}$ and $N_{dst}$.

3. Transfer ownership of $C_{migr}$ from $N_{src}$ to $N_{dst}$.

4. Notify the query router that all future transactions must be routed to $N_{dst}$. 
DEF 1.

Safe Migration. A migration technique is safe if the following conditions are met:

1. Data Safety and Unique ownership: The persistent image of a cell is consistent and only a single DBMS node owns a cell at any instant of time

2. Durability: Updates from committed transactions are durable.
GUARANTEE 1.

- Atomicity of handover. In spite of $C_{\text{migr}}$ is owned by exactly one of $N_{\text{src}}$ or $N_{\text{dst}}$.

GUARANTEE 2.

- Changes made by aborted transactions are neither persistently stored nor copied over during migration.

GUARANTEE 3.

- Changes made by committed transactions are persistent and never lost during migration.
**GUARANTEE 4.**

- Migrating active transactions does not violate the durability condition even if the commit log at $N_{src}$ is discarded after successful migration.

**GUARANTEE 5.**

- **Serializability.** Copying the transaction state in the final handover phase of Albatross ensures serializable transaction execution after migration.
**Comparison**

* **Stop and Migrate (S&M)**
  * In stop and migrate, a long unavailability window results from flushing cached updates from committed transactions.

* **Flush and Migrate (F&M)**
  * Performs a flush while continuing to serve transactions, followed by the final stop and migrate step.
EXPERIMENTAL RESULTS

(a) Transaction latency distribution (box and whisker plot). Inset shows the same graph but with a limited value of the y-axis to show the box corresponding to the $25^{th}$ and $75^{th}$ percentiles.

(b) Transaction latency observed for different migration techniques. There are 3 different series and each correspond to an execution using one of the discussed migration techniques. All series are aligned at the time at which migration was initiated (about 38 seconds). Other vertical lines denote time instances when migration completed.
Albatross results in only 5 – 15% transaction latency increase (over 80 – 100 ms average latency) in the 30 second window after migration, while both F&M and S&M result in 300–400% latency increase.
Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments
AUTHORs

* Kejiang Ye, Xiaohong Jiang, Dawei Huang, Jianhai Chen, and Bei Wang
  * College of Computer Science, Zhejiang University
This paper focuses on the live migration strategy of multiple virtual machines with different resource reservation methods.
Figure 1. Resource Reservation Based Live Migration of Multiple Virtual Machines: (a) Live Migration Framework of Multiple Virtual Machines with Resource Reservation Technology. The VM in dark represents the reserved resources in the form of virtual machines in the target machines, the CPU and Mem in dark represents the reserved resources in the source machine; (b) An Example of Parallel Live Migration of Multiple Virtual Machines.
Migration Decision-Maker

- Responsible for making effective migration decision.
- Some simple migration strategies are included in this module
  - E.g. sequential migration, parallel migration, workload-aware migration strategies

Migration Controller

- Controls the real migration process.
- Choose the right target machine from the candidate target machine list and trigger the migration at particular time.
* **Resource Reservation Controller**
  * Implements different resources reservation strategies for both source machine and target machine.
  * Avoid migration failures because of the insufficient resources.

* **Resource Monitor**
  * Monitoring the resource status of both virtual machines and physical machines, including the resource utilization, virtual machine configuration information.
    * E.g. workload characteristics, VCPU number, memory size, image size.
  * Analyze the workload stability to avoid migration thrashing.
The Resource Reservation Methods

* Resource Reservation in Source Machine
  
  * CPU resource reservation
    * Adjusting the CAP value in Xen
  
  * Memory resource reservation
    * Dynamically adjusting virtual machine memory size.
Resource Reservation in Target Machine

**CPU resource reservation**
- Create a certain number of virtual machines running CPU-bound workloads inside to achieve the goal of CPU resource reservation.

**Memory resource reservation**
- Creating one or multiple virtual machines allocated a certain amount of memory resource.
There are usually three typical metrics to quantify the performance or efficiency of live migration technologies.

- **Downtime**
  - The time that the migrating virtual machine is stopped with no response.

- **Total migration time**
  - The period from migrating start to finish.

- **Workload performance overheads**
  - The performance degradation of the running virtual machine workloads because extra machine resources are consumed to perform the migration.
Performance Overheads of Live Migration of Single VM

Table I
THE WORKLOAD PERFORMANCE OVERHEADS OF LIVE MIGRATION OF VIRTUAL MACHINES

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Migration</th>
<th>Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECjbb(bops)</td>
<td>16195</td>
<td>14853</td>
<td>8.29%</td>
</tr>
<tr>
<td>Database OLTP(s)</td>
<td>66.14</td>
<td>88.14</td>
<td>33.26%</td>
</tr>
<tr>
<td>Disk I/O Write(Kb/s)</td>
<td>450863</td>
<td>415080</td>
<td>7.93%</td>
</tr>
<tr>
<td>Web I/O(Bytes/sec)</td>
<td>1991748</td>
<td>1413509</td>
<td>29.03%</td>
</tr>
</tbody>
</table>

Table II
THE DOWNTIME AND MIGRATION TIME OF DIFFERENT VIRTUAL MACHINE WORKLOADS UNDER LIVE MIGRATION

<table>
<thead>
<tr>
<th></th>
<th>Downtime (ms)</th>
<th>Migration Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle VM</td>
<td>16</td>
<td>47.53</td>
</tr>
<tr>
<td>SPECjbb</td>
<td>460</td>
<td>66.39</td>
</tr>
<tr>
<td>Sysbench</td>
<td>18</td>
<td>118.75</td>
</tr>
<tr>
<td>IOzone</td>
<td>19</td>
<td>59.51</td>
</tr>
<tr>
<td>Webbench</td>
<td>27</td>
<td>53.91</td>
</tr>
</tbody>
</table>
Resource Reservation in Source Machine

Figure 2. CPU Resource Reservation in Source Machine when Migrating Idle Virtual Machine.

Figure 3. Memory Resource Reservation in Source Machine when Migrating Idle Virtual Machine.
Resource Reservation in Source Machine

Figure 4. CPU Resource Reservation in Source Machine when Migrating SPECjbb Virtual Machine.
Resource Reservation in Target Machine

Figure 5. CPU Resource Reservation in Target Machine by Creating CPU-Bound Virtual Machines.

Figure 6. Memory Resource Reservation in Target Machine by Creating Idle Virtual Machines with Particular Memory Size.
Parallel Live Migration of Multiple Virtual Machines

Figure 8. The Performance of Parallel Migration From 4-core Machine to 8-core Machine.

Figure 9. The Performance of Parallel Migration From 8-core Machine to 4-core Machine.
### Table III

**WORKLOAD-AWARE LIVE MIGRATION STRATEGIES PERFORMANCE COMPARISON OF SPECjbb, IOZONE, SYSBENCH, AND WEBBENCH WHEN MIGRATING TO EACH OTHER VIRTUAL MACHINES.**

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Mig. to Jbb VM</th>
<th>Mig. to File VM</th>
<th>Mig. to Database VM</th>
<th>Mig. to Web VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECjbb (bops)</td>
<td>14849</td>
<td>11996</td>
<td>13770</td>
<td>14069</td>
<td>12820</td>
</tr>
<tr>
<td>IOzone (Kbytes/sec)</td>
<td>414560</td>
<td>243604</td>
<td>109350</td>
<td>142072</td>
<td>213602</td>
</tr>
<tr>
<td>Sysbench (s)</td>
<td>88.10</td>
<td>110.96</td>
<td>126.86</td>
<td>97.04</td>
<td>117.32</td>
</tr>
<tr>
<td>Webbench (Pages/min)</td>
<td>1994840</td>
<td>1565262</td>
<td>1677740</td>
<td>1807098</td>
<td>1115668</td>
</tr>
</tbody>
</table>
**Conclusion**

- Live migration of VM workloads bring some performance overheads due to the unavailable service in the downtime and the resource competition between virtual machine workloads and the migration process.

- The performance overheads of live migration is affected by memory size, CPU resource, and the workload types.

- Although the resource reservation in target machine will not help improve the migration efficiency, it is necessary to avoid the migration failures due to the possibility of insufficient resources in the target machine.
The adequate system resources in the source machine can make more parallel number of migrations and can obtain better migration efficiency.

The performance of migrated virtual machine workloads can be affected by the **workload characteristics** of the existing virtual machines running in the target machine.
Optimization Method

* **Optimization in the source machine**
  * Dynamically adjusting CPU and memory resources in the source machine.
  * Adjusting the migration sequence that letting the VMs with small memory migrated first.

* **Parallel migration of multiple VMs**
  * With sufficient system resources in the source machine performance will be better.

* **Workload-aware migration strategy**
  * Migrate the VM workloads according to the VM workload characteristics running in the target machine.
THANK YOU!

The End …
Q: What is live migration and please list three benefits it brings.

A:
(a) Live migration allows a server administrator to move a running virtual machine or application between different physical machines without disconnecting the client or application.
(b) Energy saving, Load balancing and Online maintenance.