On Selecting the Right Optimizations for Virtual Machine Migration

Senthil Nathan N, Umesh Bellur, Purushottam Kulkarni

Systems and Networks Group (SynerG)
Department of Computer Science and Engineering
Indian Institute of Technology Bombay

April 2, 2016
Virtual Execution Environments (VEE) 2016
VM is executing: Memory pages are transferred iteratively
Pre-Copy Live Migration

VM is executing: Memory pages are transferred iteratively

source PM

VM

memory pages

copying VM's main memory

Storage Area Network

Stores VM's disk image

destination PM

VM

iter #1

migration time

iter #2

iter #n

downtime

source PM

stop VM's
execution

VM

pages dirtied
in iter #1

pages dirtied
in iter #(n-1)

remaining pages &
hardware state

destination PM

iter #1

iter #2

iter #n

stop VM's
execution

VM

migration time

downtime
Pre-Copy Live Migration

VM is executing: Memory pages are transferred iteratively

```
source PM

VM

VM's main memory

memory pages

iter #1

migration time

pages dirtied
in iter #1

pages dirtied
in iter #(n-1)

remaining pages &
hardware state

destination PM

VM

downtime

stop VM's execution

VM hosting file server = 5 minutes
```

migration time(VM hosting file server) = 5 minutes
Existing Optimization Components for Pre-Copy Live Migration

Page Dirty Characteristics
1. page delta (del).
2. frequently/hot dirty pages (hdp).

Page Content Characteristics
1. zero pages (zp) & sub zero pages (szp).
2. duplicate pages (dup) & sub duplicate pages (sdup).
3. word-level duplicates (wld).

We identified a new component: False Dirty Page (fdp)
False Dirty Pages (fdp)

Before Starting an Iteration

- User Space Migration Process
  - retrieve dirty bitmap
  - reset dirty bitmap & mark all pages of VM as read only

During an Iteration

- Kernel Space Hypervisor
  - Transfer pages that are marked in the received dirty bitmap

1. write VM’s read only page
2. set the dirty bit
3. mark the page writable

1. An iteration can transfer a page that is **already dirted** during current iteration.

2. What happens if this page **never get dirtied again** before the next iteration?
   - An unmodified page is transferred: False Dirty Page.
Existing Optimizations

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Account for</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dirty</td>
<td>content</td>
</tr>
<tr>
<td></td>
<td>characteristics</td>
<td>characteristics</td>
</tr>
<tr>
<td></td>
<td>fdp</td>
<td>del</td>
</tr>
<tr>
<td>Delta Compression</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Page Skip</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Full Page Deduplication</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sub Page Deduplication</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Data Compression</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

False Dirty Page (fdp) is overlooked in existing study

Question of Interest

1. Which is the best technique in terms of performance & cost?
2. Does optimization’s performance depend on application behaviour?
3. What combinations of optimizations are more appropriate?
4. Can the combinations of optimizations reduce the resource cost?

fdp helps in answering these questions
ALL PAGES are scheduled to transfer in iter #1

Iter. #1

Send List

Collect n pages & transfer  
locate all pages dirtied during this transfer time
ITER. #1
---------

ALL PAGES are scheduled to transfer in iter #1

Send List

Collect n pages & transfer

locate all pages dirtied during this transfer time

Send List

\[ \bullet \] denotes dirtied pages during the transfer of 'n' pages

Collect n pages & transfer

dirtied pages yet to be transferred are skipped
Page Skip

**ITER. #1**

ALL PAGES are scheduled to transfer in iter #1

Send List

Collect n pages & transfer

locate all pages dirtied during this transfer time

\[\text{\bullet \ denotes dirtied pages during the transfer of 'n' pages}\]

\[\text{\bullet \ denotes dirtied pages during the transfer of 'n' pages}\]

Collect n pages & transfer

\[\text{dirtied pages yet to be transferred are skipped}\]

Send List

Collect <=n pages & transfer

**ITER. #2**

ALL dirtied PAGES during iter #1

Send List

Collect n pages & transfer

\[\text{\bullet \ denotes dirtied pages during the transfer of 'n' pages}\]

\[\text{\bullet \ denotes dirtied pages during the transfer of 'n' pages}\]

\[\text{Eligible pages for skip: Pages dirtied in both iteration 1 and 2}\]

Dirted pages during the transfer of 'n' pages

\[\text{...... \bullet \ ...... \bullet \ ...... \bullet \ ......}\]
Our 3 Goals

1. To perform a comprehensive empirical study to understand and compare between performance and cost of different migration optimization techniques.

2. To understand the relationship between application behavior and the payback of different optimizations.

3. To investigate the feasibility of combining optimizations so as to get the maximum improvement in migration performance for unit resource cost.
Experimental Setup and Workloads Used for Evaluation

**Table: Workloads: Web Services**

<table>
<thead>
<tr>
<th>Workloads</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mediawiki, DVD Store, RUBiS</td>
<td>webserver and database server</td>
</tr>
<tr>
<td>OLTPBench</td>
<td>epinions, twitter, seats, voter, tpcc, tatp, ycsb</td>
</tr>
<tr>
<td>HTTP file server</td>
<td>100s of files each with different sizes</td>
</tr>
</tbody>
</table>

**Table: Workloads: Multimedia, Biotech, Mining, MPI**

<table>
<thead>
<tr>
<th>Workloads</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc</td>
<td>kernel compile with multiple threads</td>
</tr>
<tr>
<td>File compression</td>
<td>compress 8 GB data set with tar</td>
</tr>
<tr>
<td>Linpack</td>
<td>linear equation solver</td>
</tr>
<tr>
<td>Nu-Minebench (data mining)</td>
<td>ECLAT, HOP, ScalParC, UtilityMine</td>
</tr>
<tr>
<td>Mummer</td>
<td>gnome sequence alignment</td>
</tr>
<tr>
<td>Parsec (multimedia tools)</td>
<td>bodytrack, ferret, fluidanimate, freqmine, vips, and x264</td>
</tr>
<tr>
<td>Dacapo (java benchmarks)</td>
<td>avrora, eclipse, fop, h2, jython, luindex, lusearch, pmd, sunflow, tomcat, tradebeans, tradesoap, and xalan</td>
</tr>
</tbody>
</table>

42 applications, 7 different migration rates.  
42 × 7 = 294 measured values.
Expectations on Optimization Performance

Performance of **Delta Compression & Page Skip** $\propto$ **Additional Network Traffic Generated**

Performance of **Deduplication** $\propto$ Additional Network Traffic & the Amount of Zero, Duplicate Pages

Performance of **Data Compression** $\propto$ Total Network Traffic & the Amount of Zero, Word Level Dup.
1. Apply each optimization individually on 294 migration instances.

2. If expectation fails, identify the reason (i.e., application behavior).

3. Compare performance & cost of each optimization and find appropriate combinations.
## Goal 1: Performance and Cost Trade-Off

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Avg. traffic reduction</th>
<th>Increase in CPU at source PM</th>
<th>Increase in CPU at destination PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Skip</td>
<td>20 %</td>
<td>1 x</td>
<td>1 x</td>
</tr>
<tr>
<td>Data Compression</td>
<td>37 %</td>
<td>5 x</td>
<td>1.3 x</td>
</tr>
<tr>
<td>Delta Compression</td>
<td>17 %</td>
<td>3.2 x</td>
<td>1.2 x</td>
</tr>
<tr>
<td>Full Page Dedup.</td>
<td>17 %</td>
<td>11 x</td>
<td>1 x</td>
</tr>
<tr>
<td>Sub Page Dedup.</td>
<td>20 %</td>
<td>13 x</td>
<td>1.2 x</td>
</tr>
</tbody>
</table>

Page skip is the best optimization in terms of performance-cost tradeoff
Observations: Reduction in network traffic

- Reduction due to data compression is significant.
- On average, 40% reduction due to False Dirty Pages (fdp).
- Approximately, 53% reduction due to Hot Dirty Pages (hdp).
Goal 2: Co-relation Between Application Behavior and Optimization’s Performance

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Low performance when</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta Compression</td>
<td>high dirty frequency, high disk read, memset(), free()</td>
</tr>
<tr>
<td>Page Skip</td>
<td>page dirty rate &lt;&lt; migration rate and large writable set size</td>
</tr>
<tr>
<td>Deduplication</td>
<td>high dirty frequency and low dup+zp</td>
</tr>
<tr>
<td>Data Compression</td>
<td>low word level duplicates &amp; zero content</td>
</tr>
</tbody>
</table>
Delta Compression: Impact of Application Behaviour
Δelta Compression: Impact of Application Behaviour

\[ \frac{\text{disk read}, \text{memset()}, \text{free()}}{\text{modified all 4 KB}} \]

- \( fdp \propto \frac{1}{d_f} \)
- high \( d_f \) \( \implies \) improvement dependent on page modification size.

IIT Bombay
### Goal 3: Appropriate Combinations—Page Skip as Base Optimization

Page Skip + Data Compression is the best combination in terms of performance-cost tradeoff.
Reduction in Total CPU Utilization due to Combination of Techniques

(a) Individual Technique

(b) Page skip as the base technique
Reduction in Total CPU Utilization due to Combination of Techniques

(a) Individual Technique

(b) Page skip as the base technique
Reduction in Impact of Application Behaviour due to Combination of Techniques

<table>
<thead>
<tr>
<th>Page Skip +</th>
<th>low performance gain when</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delta Compression</strong></td>
<td>high-dirty-frequency, unskipped pages are dirtied due to disk read, memset(), free()</td>
</tr>
<tr>
<td><strong>Deduplication</strong></td>
<td>high-dirty-frequency, low dup+zp</td>
</tr>
<tr>
<td><strong>Data Compression</strong></td>
<td>low word level duplicates and low zero content</td>
</tr>
<tr>
<td>Page Skip + Data</td>
<td>Reduction in traffic</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Compression +</td>
<td>over page skip + data compression</td>
</tr>
<tr>
<td>Delta Compression</td>
<td>3 %</td>
</tr>
<tr>
<td>Full Page Dedup.</td>
<td>2 %</td>
</tr>
<tr>
<td>Sub Page Dedup.</td>
<td>-3 %</td>
</tr>
</tbody>
</table>

No more suitable combinations
More Results (refer to paper)

1. Reduction in migration time and downtime due to each technique.
2. Detailed CPU utilization with respect to migration rate.
3. Impact of optimization parameters on performance improvement.
   - cache size for delta compression.
   - build frequency for page skip.
   - sub page size for deduplication.
4. Proposed a new cache replacement policy for delta compression that improves the performance over 30%.
Summary of Key Results

1. Page Skip is the best optimization (must apply).

2. Page Skip + Data Compression is the best combination.

3. Deduplication technique need to applied with utmost care (can use KSM).

4. Combination of techniques reduce the
   - impact of application behaviour on optimization’s performance.
   - total resource utilization (while increasing per second utilization).
Thank You

We thank Google & Microsoft Research India for travel grant

Questions?