Tesseract: Reconciling Guest I/O and Hypervisor Swapping in a VM

Kapil Arya  
Northeastern University

Yury Baskakov  
VMware, Inc.

Alex Garthwaite  
Twitter, Inc.

March 1, 2014
Host swapout followed by guest disk read
Host swapout followed by guest overwriting the entire page (e.g. zeroing)
Host swap out of an unmodified guest page
Host swapout followed by guest disk write

*Double-Paging*
VSwapper (Amit et al., ASPLOS’14) describes some more cases
Hypervisor Swapping and Guest I/O: Sub-Optimal Behavior

Host swapout followed by guest disk read
(stale swap reads)

Host swapout followed by guest overwriting the entire page
(false swap reads)

Host swap out of an unmodified guest page
(slient swap writes)

Host swapout followed by guest disk write

- **VSwapper** (Amit et al., ASPLOS’14) describes some more cases
- **Tesseract** focuses on the *Double-Paging* problem
  - our baseline has solutions for stale-swap-reads and false-swap-reads
The Double Paging Problem

- Goldberg/Hassinger 1974; Seawright/MacKinnon 1979
- Memory overcommitment at both host level and guest level
- Guest paging of page swapped out by hypervisor
- Swap in contents only to write it out again
- Existing solutions:
  - Specialized guest paging device (Govil et al. 1999)
  - Paravirtualization (Geiger: Jones 2002; Lu and Shen 2007)
Two Levels of Memory Scheduling

LP: (guest) Logical Page;  PPN: Physical Page Number;  MPN: Machine Page Number;
(1) Remove PPN to MPN mapping

Arya, Baskakov, Garthwaite
(1) Remove PPN to MPN mapping

(2) Write MPN to host paging device;

---

**Figure:** Diagram illustrating the process of Host Level Swap Out. The diagram shows the mapping between Guest Physical Memory (PPN) and Host Memory (MPN). The process involves removing the PPN to MPN mapping and then writing the MPN to the host paging device.
(1) Remove PPN to MPN mapping
(2) Write MPN to host paging device; reuse MPN
Guest Level Swap Out: Resulting in a Double Paging

1. Remove PPN to MPN mapping
2. Write MPN to host paging device; reuse MPN
3. Guest block write request
(1) Remove PPN to MPN mapping
(2) Write MPN to host paging device; reuse MPN
(3) Guest block write request
(4) Memory allocation and swap in
Guest Level Swap Out: Resulting in a Double Paging

1. Remove PPN to MPN mapping
2. Write MPN to host paging device; reuse MPN
3. Guest block write request
4. Memory allocation and swap in
5. Establish PPN to MPN mapping
Guest Level Swap Out: Resulting in a Double Paging

1. Remove PPN to MPN mapping
2. Write MPN to host paging device; reuse MPN
3. Guest block write request
4. Memory allocation and swap in
5. Establish PPN to MPN mapping
6. Write block to guest disk
Guest Level Swap Out: Resulting in a Double Paging

1. Remove PPN to MPN mapping
2. Write MPN to host paging device; reuse MPN
3. Guest block write request
4. Memory allocation and swap in
5. Establish PPN to MPN mapping
6. Write block to guest disk
7. Zero the new MPN for reuse
Solution: Block-Swap Store (BSST)

- Guest VMDK
- BSST Block Mapping Info
- Guest Physical Memory
  - guest view
  - hypervisor view
- Host Memory
- MPN
- PPN

LP1

Arya, Baskakov, Garthwaite

Tesseract

March 1, 2014
Solution: Block-Swap Store (BSST)

PPN to BSST mapping

<table>
<thead>
<tr>
<th>PPN</th>
<th>BSST</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>&lt; MPN 1 &gt;</td>
<td>&lt; bBlock1 &gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Solution: Block-Swap Store (BSST)

Guest VMDK to BSST indirection

<table>
<thead>
<tr>
<th>guest block</th>
<th>BSST</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; gBlock1 &gt;</td>
<td>&lt; bBlock1 &gt;</td>
</tr>
</tbody>
</table>

PPN to BSST mapping

<table>
<thead>
<tr>
<th>PPN</th>
<th>BSST</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; MPN 1 &gt;</td>
<td>&lt; bBlock1 &gt;</td>
</tr>
</tbody>
</table>

LP1

Guest Physical Memory

guest view

hypervisor view

Host Memory

hypervisor swapping

PPN

guest I/O

Arya, Baskakov, Garthwaite

Tesseract

March 1, 2014
Solution: Block-Swap Store (BSST)

Guest
VMDK
BSST
Block Mapping Info
hypervisor
swapping

LP1
Guest Physical Memory
Host Memory

PPN
MPN

guest I/O
guest view
hypervisor view

VMM SwapOut
Allocate Memory
Synchronous SwapIn
Guest Write I/O
Zero Write
Update PTE

Write Metadata
PShare
Update PTE

guest VMDK to BSST indirection

<table>
<thead>
<tr>
<th>guest block</th>
<th>BSST</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; gBlock1 &gt;</td>
<td>&lt; bBlock1 &gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

PPN to BSST mapping

<table>
<thead>
<tr>
<th>PPN</th>
<th>BSST</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; MPN 1 &gt;</td>
<td>&lt; bBlock1 &gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Explicit swapper and block-swap store VMDK (BSST)

Track associations between memory pages and disk blocks

Allow indirections between guest VMDKs and BSST

Defragment guest VMDKs
Select Cold Pages for Swapping

- The existing hypervisor swapper avoids picking pages the guest OS is likely to page out
  - this is done through random selection

Tesseract doesn't suffer from double-paging; it is good if the guest and hypervisor both page out the same set of swap-out coldest pages using random page selection with a victim cache.
The existing hypervisor swapper avoids picking pages the guest OS is likely to page out.
- This is done through random selection.

Tesseract doesn’t suffer from double-paging:
- It is good if the guest and hypervisor both page out the same set of pages.
- Swap out coldest pages.
- Uses random page selection with victim cache.
Explicit Page Swapper

- Choose cold(er) pages for swapping
- Create PPN to BSST mappings on swap out
- Intercept writes to memory to update mappings
  - guest disk reads
  - zeroing of page
  - cancel swap-in if the entire page is being overwritten

Arya, Baskakov, Garthwaite
Tesseract
March 1, 2014 10 / 23
Guest VMDK to BSST Mappings: Guest Write Request

1. Original scatter-gather list
2. Tesseract intercepts guest write requests and finds swapped pages
3. Rewrite scatter-gather lists based on indirections to avoid double-paging
4. Establish guest VMDK to BSST mappings

VMware workstation I/O stack:

- Guest Operating System
- Virtual Machine Monitor (VMM)
- SCSI Disk Device
  - I/O dispatch
  - I/O completion
- Asynchronous I/O Manager
- Host File Layer

VMX

Asynchronous I/O Manager

Tesseract

VMX
Guest VMDK to BSST Mappings: Guest Write Request

Write request issued:

1. Original scatter-gather list

   1 2 3 4 5 6 7 8

2. Tesseract intercepts guest write requests and finds swapped pages

   1 2 3 4 5 6 7 8

3. Rewrite scatter-gather lists based on indirections to avoid double-paging

   /0/0/0
   /0/0/0
   /0/0/0
   /1/1/1
   /1/1/1
   /1/1/1
   /0/0
   /0/0
   /0/0
   /1/1
   /1/1
   /1/1
   /0/0/0
   /0/0/0
   /0/0/0
   /1/1/1
   /1/1/1
   /1/1/1
   1 3 5 8

4. Establish guest VMDK to BSST mappings

   VMware workstation I/O stack

   - Guest Operating System
   - Virtual Machine Monitor (VMM)
   - SCSI Disk Device
     - I/O dispatch
     - Asynchronous I/O Manager
     - I/O completion
   - VMX
   - Host File Layer
Guest VMDK to BSST Mappings: Guest Write Request

Write request issued:

1. Original scatter-gather list
   
   1 2 3 4 5 6 7 8

2. Tesseract intercepts guest write requests and finds swapped pages
   
   1 2 3 4 5 6 7 8

VMware workstation I/O stack:

1. Guest Operating System
2. Virtual Machine Monitor (VMM)
3. SCSI Disk Device
4. Asynchronous I/O Manager
5. I/O dispatch
6. I/O completion

VMX

Host File Layer

Asynchronous I/O Manager

I/O dispatch

I/O completion

Virtual Machine Monitor (VMM)

Guest Operating System
Guest VMDK to BSST Mappings: Guest Write Request

Write request issued:

(1) Original scatter-gather list

(2) Tesseract intercepts guest write requests and finds swapped pages

(3) Rewrite scatter-gather lists based on indirections to avoid double-paging

VMware workstation I/O stack

Guest Operating System

Virtual Machine Monitor (VMM)

SCSI Disk Device

I/O dispatch

I/O completion

Asynchronous I/O Manager

VMX

Host File Layer

Arya, Baskakov, Garthwaite
Guest VMDK to BSST Mappings: Guest Write Request

Write request issued:

(1) Original scatter-gather list

1 2 3 4 5 6 7 8

(2) Tesseract intercepts guest write requests and finds swapped pages

1 2 3 4 5 6 7 8

(3) Rewrite scatter-gather lists based on indirections to avoid double-paging

1 3 5 8

Write completion phase:

(6) Establish guest VMDK to BSST mappings
Read request issued:

(1) Original scatter-gather list

1 2 3 4 5 6 7 8

VMware workstation I/O stack

Guest Operating System

Virtual Machine Monitor (VMM)

SCSI Disk Device

I/O dispatch

I/O completion

Asynchronous I/O Manager

Host File Layer
Guest VMDK to BSST Mappings: Guest Read Request

Read request issued:

(1) Original scatter-gather list

1 2 3 4 5 6 7 8

(2) lookup guest VMDK indirections

1 2 3 4 5 6 7 8

VMware workstation I/O stack

Guest Operating System

(1)

Virtual Machine Monitor (VMM)

(3)

SCSI Disk Device

(2) (6)

I/O dispatch

I/O completion

(5)

Asynchronous I/O Manager

(4)

Host File Layer

VMX
Guest VMDK to BSST Mappings: Guest Read Request

Read request issued:

1. Original scatter-gather list
   1 2 3 4 5 6 7 8

2. Lookup guest VMDK indirections
   1 2 3 4 5 6 7 8

3. Split original scatter-gather list
   1 3 5 8 2 4 6 7

VMware workstation I/O stack

- Guest Operating System
  (1)
- Virtual Machine Monitor (VMM)
  (1)
- SCSI Disk Device
  (2) (6)
- I/O dispatch
- I/O completion
  (6)
- Asynchronous I/O Manager
  (3)
- VMX
- Host File Layer
  (4)
  (5)

Arya, Baskakov, Garthwaite

March 1, 2014 12 / 23
Guest VMDK to BSST Mappings: Guest Read Request

Read request issued:

(1) Original scatter-gather list

1 2 3 4 5 6 7 8

(2) lookup guest VMDK indirections

1 2 3 4 5 6 7 8

(3) Split original scatter-gather list

1 2 3 4 5 6 7 8

Read completion:

(6) Wait for all completion events and notify the guest.

VMware workstation I/O stack

Guest Operating System

Virtual Machine Monitor (VMM)

SCSI Disk Device

I/O dispatch

I/O completion

Asynchronous I/O Manager

Host File Layer

VMX
BSST Side Effects – Fragmented Guest VMDKs!

- Trade-off between write-prepare (2) and read-completion (6) times.
- Increased seek times.
- Increased load on disks.

VMware workstation I/O stack

Guest Operating System

Virtual Machine Monitor (VMM)

SCSI Disk Device

I/O dispatch

I/O completion

Asynchronous I/O Manager

Host File Layer

Guest VMDK

BSST VMDK

VMX

Arya, Baskakov, Garthwaite

Tesseract

March 1, 2014 13 / 23
Guest Defragmentation

Copy blocks from BSST VMDK to guest VMDK in background.

In almost all cases, background defragmentation completes before the corresponding guest requests arrive.
Guest Defragmentation

- Copy blocks from BSST VMDK to guest VMDK in background
Guest Defragmentation

- Copy blocks from BSST VMDK to guest VMDK in background
Guest Defragmentation

- Copy blocks from BSST VMDK to guest VMDK in background
- Remove guest block to BSST block mapping
- In almost all cases, background defragmentation completes before the corresponding guest requests arrive
Challenges in Evaluating Performance

- How to deterministically introduce double-paging activity?
  - hypervisor memory-limit set on VM
  - memhog customized to map/pin pages inside the VM
  - inflation with pinning followed by inflation without pinning
  - first forces hypervisor to swap pages; second induces guest paging

- How to correlate double-paging with application performance?
  - VM responsiveness
  - modified SpecJBB2005 benchmark that emits instantaneous scores
  - application pauses observed in the instantaneous scores emitted by SpecJBB
Experimental Setup

- **Host setup**
  - AMD Opteron 6168 (Magny-Cours) with 12 1.9 GHz cores
  - 1.5 GB RAM
  - 64-bit OpenSUSE 11.4

- **Guest setup:**
  - 6 VCPUs
  - 700 MB RAM
  - 64-bit Ubuntu 11.04

- **SpecJBB2005 Benchmark:**
  - 6 warehouses
  - 120 second run

- Unless specified otherwise, test runs are with 10% host overcommitment and 60 MB memhog inside the guest
Gaps in the per-second logged instantaneous scores
Memhog size: 60 MB
Application Performance with Defragmentation

![Graph showing application performance with defragmentation. The graph plots SPECjbb score against total SPECjbb blockage time (seconds). Different markers represent different conditions: baseline, no-defrag, bsst-defrag, and guest-defrag.]
## Time Spent in various I/O Paths

Average read and write prepare/completion times (in microseconds)

<table>
<thead>
<tr>
<th>I/O Path</th>
<th>Baseline</th>
<th>Tesseract</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>No-defrag</td>
<td>Guest defrag</td>
</tr>
<tr>
<td>Write prepare</td>
<td>24,262</td>
<td>220</td>
<td>265</td>
<td></td>
</tr>
<tr>
<td>Write completion</td>
<td>0</td>
<td>49</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>Read prepare</td>
<td>0</td>
<td>37</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>Read completion</td>
<td>0</td>
<td>232</td>
<td></td>
<td>55</td>
</tr>
</tbody>
</table>
Memhog size was 60 MB.
Conclusion and Future Directions

- Scatter-gather lists too closely tied to the traditional physical disks
- Extend to guest-write followed by swap scenario
- Also extend to deduplicate guest I/Os following guest I/Os
  - complete I/O deduplication with sea-of-blocks containing all data blocks
  - guest VMDKs simply contain indirections to the sea-of-blocks
- Throttle defragmentation according to disk load
  - VMware ESX throttles disk I/O (Manning and Dieckhans 2010)