SmartMD: A High Performance Deduplication Engine with Mixed Pages

Fan Guo¹, Yongkun Li¹, Yinlong Xu¹, Song Jiang², John C. S. Lui³

¹University of Science and Technology of China
²University of Texas, Arlington
³The Chinese University of Hong Kong
Overview

➢ **TLB (Translation Lookaside Buffer) miss** carries high penalty
   - Due to the access of page table
   - E.g., four level address mapping for x86-64 system with 4KB-page memory

➢ **Virtualization** increases TLB miss penalty
   - 2D page table walk (GVA -> GPA -> HPA)
   - Up to 24 memory references

➢ Uneven increase of TLB & memory size exacerbates the problem
Large Pages

Large pages improve memory access performance

- Fewer page table entries (1/512)
- Larger TLB coverage
Benefits of Large Pages

- Performance improvement with large page
  - Enabling large pages in both guest and host can improve memory access performance by up to 68%

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Host: Base Guest: Large</th>
<th>Host: Large Guest: Base</th>
<th>Host: Large Guest: Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECjbb</td>
<td>1.06</td>
<td>1.12</td>
<td>1.30</td>
</tr>
<tr>
<td>Graph500</td>
<td>1.26</td>
<td>1.34</td>
<td><strong>1.68</strong></td>
</tr>
<tr>
<td>Liblinear</td>
<td>1.13</td>
<td>1.14</td>
<td>1.37</td>
</tr>
<tr>
<td>Sysbench</td>
<td>1.07</td>
<td>1.09</td>
<td>1.20</td>
</tr>
<tr>
<td>Biobench</td>
<td>1.02</td>
<td>1.18</td>
<td>1.37</td>
</tr>
</tbody>
</table>
Deduplication with Large Pages

➢ Redundant data is very common among VMs
  - Many base pages (4KB) share the same content

➢ Large pages reduce the deduplication opportunities
  - Very few large pages (2MB) are exactly the same
  - ADA: aggressively split large pages into base pages

<table>
<thead>
<tr>
<th>Policy</th>
<th>Benchmark</th>
<th>Memory Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large Page w/o ADA</td>
<td>Graph500</td>
<td>0.37 GB (3.4%)</td>
</tr>
<tr>
<td></td>
<td>SPECjbb2005</td>
<td>0.40 GB (5.9%)</td>
</tr>
<tr>
<td></td>
<td>Liblinear</td>
<td>0.32 GB (2.0%)</td>
</tr>
<tr>
<td></td>
<td>Sysbench</td>
<td>0.09 GB (0.8%)</td>
</tr>
<tr>
<td></td>
<td>Biobench</td>
<td>0.20 GB (1.4%)</td>
</tr>
<tr>
<td>Large Page with ADA</td>
<td>Graph500</td>
<td>3.18 GB (47.9%)</td>
</tr>
<tr>
<td></td>
<td>Specjbb2005</td>
<td>1.83 GB (26.9%)</td>
</tr>
<tr>
<td></td>
<td>Liblinear</td>
<td>3.79 GB (23.7%)</td>
</tr>
<tr>
<td></td>
<td>Sysbench</td>
<td>2.83 GB (18.0%)</td>
</tr>
<tr>
<td></td>
<td>Biobench</td>
<td>1.88 GB (13.7%)</td>
</tr>
</tbody>
</table>

Dedup. with large pages (0.8% ~ 5.9%)
Dedup. with base pages (13.7% ~ 47.9%)
Base pages vs. large pages

- Exists a **tradeoff** between access performance and deduplication rate

<table>
<thead>
<tr>
<th></th>
<th>Access Performance</th>
<th>Deduplication Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base pages (4KB)</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Large Pages (2MB)</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Question:** can we enjoy **both benefits** of high access performance and high deduplication rate simultaneously?
Our Solution

➢ **SmartMD**: an adaptive management scheme with mixed pages
  - Monitors page information (access frequency, repetition rate)
  - Adaptively splits/reconstructs large pages: manage with mixed pages

➢ **Observation**: many large pages have high access frequency but few duplicate subpages
High-level Idea of SmartMD

- Lightweight scheme to monitor page information
  - Access frequency and repetition rate

- Adaptive scheme to **selectively** split/reconstruct large pages
  - **Split** into base pages
    - Cold pages with high repetition rate
    - For high deduplication rate
  - **Keep** in large pages
    - Hot pages with low repetition rate
    - For high access performance
  - **Reconstruct**: hot pages
    - For high access performance
Key Issues

- Access frequency monitor
- Repetition rate monitor
- Adaptive conversion
Monitor Access Frequency

➢ Scan pages periodically

➢ In each scan interval (e.g., 6s)
  − Reset the access bits of all pages
  − Sleep (e.g., 2.6s)
  − Check the access bits & update access frequency (+/- by one)
  − Sleep until this scan interval ends

➢ Use a counter to keep the access frequency of each large page
Monitor Repetition Rate

- Scan pages periodically, and for each large page
  - Check each of its subpages and label it if it is a duplicate
  - Use a counter to record repetition rate

- Counting bloom filter
  - # of entries: 8 # of base pages
  - Each entry: a 3-bit counter
  - 3 hash functions to index

- Sampling
  - Sample only 25% subpages for pages being checked before and not being modified in last interval
Selectively split/reconstruct: **adjust** para. based on mem. util.

- **Split**: Acc. Freq. < \text{Thres}_{\text{cold}} \& \text{Rep. Rate} > \text{Thres}_{\text{repet}}
- **Reconstruct**: Acc. Freq. > \text{Thres}_{\text{hot}}

**Implementation**

- **Split**: well supported by Linux
- **Reconstruct**
  - Gathering subpages
  - Migrate remapped subpages
  - Break shared subpages
  - Recreating page descriptor
  - Updating page table
Deduplication thread
  - Modify KSM’s deduplication algorithm to merge duplicated pages
    • Two red-black trees to manage pages
  - With duplicate labels, **SmartMD improves deduplication efficiency**
    • Compare pages with duplicate labels only
    • The # of candidate pages for comparison is reduced
    • The height of the red-black trees is reduced
    • The # of comparisons to merge a page is reduced
Evaluation

➢ Experiment setting
  - **Host**: two Intel Xeon E5-2650 v4 2.20GHz processors, 64GB RAM
  - **Guest**: QEMU&KVM. Boot up 4 VMs on one physical CPU, each VM is assigned one VCPU and 4GB RAM
  - Both guest and host OSes are Ubuntu 14.04

➢ Workloads and memory demands w/o deduplication

<table>
<thead>
<tr>
<th>Graph-500</th>
<th>SPECjbb</th>
<th>Liblinear</th>
<th>Sysbench</th>
<th>Biobench</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.7GB</td>
<td>1.7GB</td>
<td>4.0GB</td>
<td>2.93GB</td>
<td>3.42GB</td>
</tr>
</tbody>
</table>
Overhead of SmartMD

- **SmartMD** reduces CPU consumption even if it requires more CPU cycles for monitoring
  - Average CPU utilization sampled in every second

<table>
<thead>
<tr>
<th></th>
<th>Monitor thread</th>
<th>Dedup thread</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSM</td>
<td>0</td>
<td>33.5%</td>
<td>33.5%</td>
</tr>
<tr>
<td>Ingens</td>
<td>5.3%</td>
<td>21.3%</td>
<td>26.6%</td>
</tr>
<tr>
<td>SmartMD</td>
<td>13.1%</td>
<td>11.9%</td>
<td>25.0%</td>
</tr>
</tbody>
</table>

- **SmartMD** introduces negligible memory overhead
  - $3/2^{12}$ for storing counting bloom filter, and $1/2^{16}$ for keeping access frequency & repetition rate
  - Tens of MB for 16GB memory
Performance of SmartMD

➢ Comparison deduplication schemes
  – **KSM**: aggressively splits large pages which contain duplicate subpages
    • Already supported in Linux
    • Achieves best memory saving
  – **No-splitting**: deduplicates memory in unit of 2MB page
    • Without splitting any large page
    • Achieves best access performance
  – **Ingens (OSDI’16)**: Splits large pages with low access frequency w/o considering repetition rate
➢ KSM and no-splitting stand for two extreme points on the tradeoff curve (best performance vs. best memory saving)
KSM and no-splitting stand for two extreme points on the tradeoff curve (best performance vs. best memory saving)

SmartMD achieves
- similar performance with no-splitting
- similar memory saving with KSM

Takes both benefits simultaneously
Comparison with Ingens

- Memory saving

- SmartMD can save 30% to 2.5x more memory than Ingens with similar access performance
Performance in Overcommitted Systems

➢ Overcommitment level (ratios of memory demand of all VMs to usable memory size): 0.8, 1.1, 1.4
- Limit the host’s memory by running an in-memory file system (hugetlbfs)

➢ SmartMD achieves up to 38.6% of performance improvement over other schemes
Performance on NUMA Machine

Setting: 2 VMs on one physical CPU and two on a different CPU
  - Baseline: no-splitting (best access performance)

<table>
<thead>
<tr>
<th></th>
<th>Single-CPU</th>
<th>NUMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph500</td>
<td>0.8%</td>
<td>1.6%</td>
</tr>
<tr>
<td>SPECjbb2005</td>
<td>0.6%</td>
<td>2.1%</td>
</tr>
<tr>
<td>Liblinear</td>
<td>0.9%</td>
<td>1.8%</td>
</tr>
<tr>
<td>Sysbench</td>
<td>1.1%</td>
<td>2.6%</td>
</tr>
<tr>
<td>Biobench</td>
<td>1.8%</td>
<td>3.9%</td>
</tr>
</tbody>
</table>

NUMA effect is very small
  - The extra performance reduction on NUMA machine is < 2% comparing to Single-CPU
Conclusions

➢ Tradeoff: large pages improve memory access performance, but reduce deduplication opportunities
  - Many pages have high access frequency but few duplicate subpages

➢ We propose SmartMD, an adaptive scheme to manage memory with mixed pages
  - Split: cold pages with high repetition rate
  - Reconstruct: hot pages
  - SmartMD simultaneously takes both benefits
    • High memory performance (by accessing with large pages)
    • High memory saving (by deduplicating with base pages)
Thanks!

Q&A