Karlsruhe Institute of Technology

Deduplication in VM Environments

Frank Bellosa <bellosa@kit.edu>
Konrad Miller <miller@kit.edu>
Marc Rittinghaus <rittinghaus@kit.edu>

KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT) - SYSTEM ARCHITECTURE GROUP

```
2606  for (; vma && ksm_scan.address < hint->end ; vma = vma->vm_next) {
2607          if (!(vma->vm_flags & VM_MERGEABLE))
2608                  continue;
2609          if (ksm_scan.address < vma->vm_start)
2610                  ksm_scan.address = vma->vm_start;
2611          if (!vma->anon_vma)
2612                  ksm_scan.address = vma->vm_end;
2613
2614          while (ksm_scan.address < vma->vm_end && ksm_scan.address < hint->end ) {
2615                  if (ksm_test_exit(mm))
2616                          break;
2617                  *page = follow_page(vma, ksm_scan.address, FOLL_GET);
2618                  if (IS_ERR_OR_NULL(*page)) {
2619                          ksm_scan.address += PAGE_SIZE;
2620                          ksm_cond_resched();
2621                          continue;
2622                  }
2623                  if (PageAnon(*page) ||
2624                      page_trans_compound_anon(*page)) {
2625                          flush_anon_page(vma, *page, ksm_scan.address);
2626                          flush_dcache_page(*page);
2627
2628                          rmap_item = get_next_rmap_item_hint(slot,
2629                                  ksm_scan.rmap_list, ksm_scan.address);
```

Memory Sharing in VM Environments

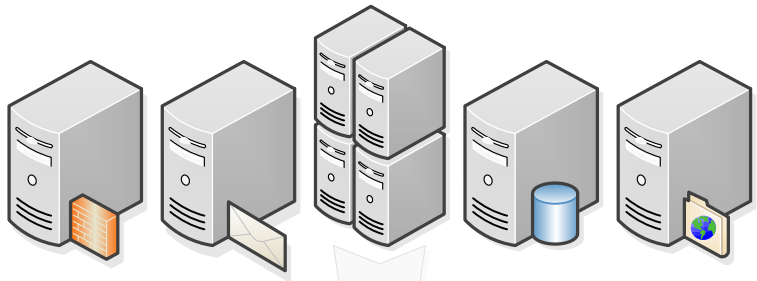
- Operating Systems do a fairly good job in sharing, but there is still duplicate content in memory
  - How much ?
  - How long ?
  - Wherefrom ?
- State of the art in deduplication
- KSM++: introducing hints for memory scanners
- Evaluation
- Challenge: NUMA and SCM

Memory Sharing in VM Environments

- Operating Systems do a fairly good job in sharing, but there is still duplicate content in memory
  - How much ?
  - How long ?
  - Wherefrom ?
- State of the art in deduplication
- KSM++: introducing hints for memory scanners
- Evaluation
- Challenge: NUMA and SCM

Virtualization for Server Consolidation

- *Past*:
  One physical machine for each service

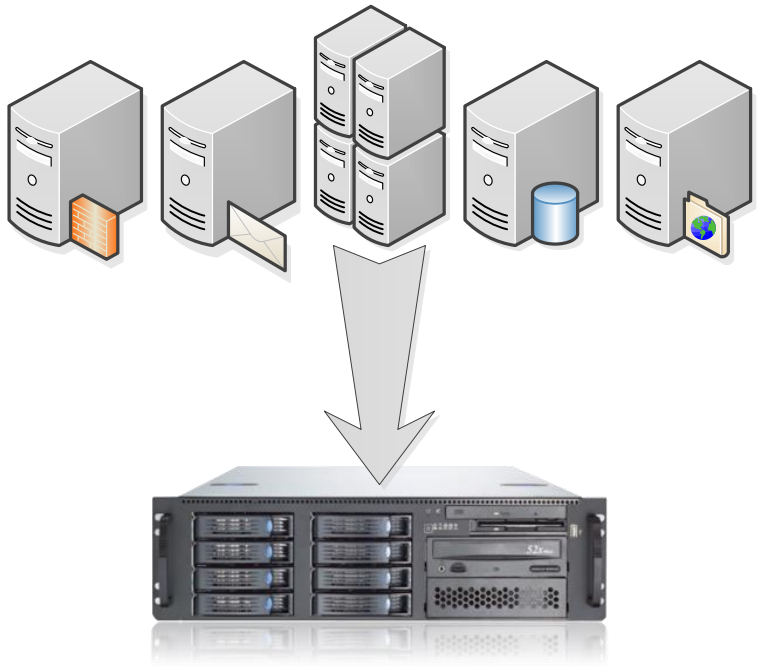


- *Present*:
  Multiple isolated virtual machines on a single physical host
  - Improved hardware utilization
  - Increased flexibility (placement/migration)
  - Smaller hardware footprint
  - Energy efficiency

Virtualization for Server Consolidation

- *Past*:
  One physical machine for each service

- *Present*:
  Multiple isolated virtual machines on a single physical host
  - Improved hardware utilization
  - Increased flexibility (placement/migration)
  - Smaller hardware footprint
  - Energy efficiency



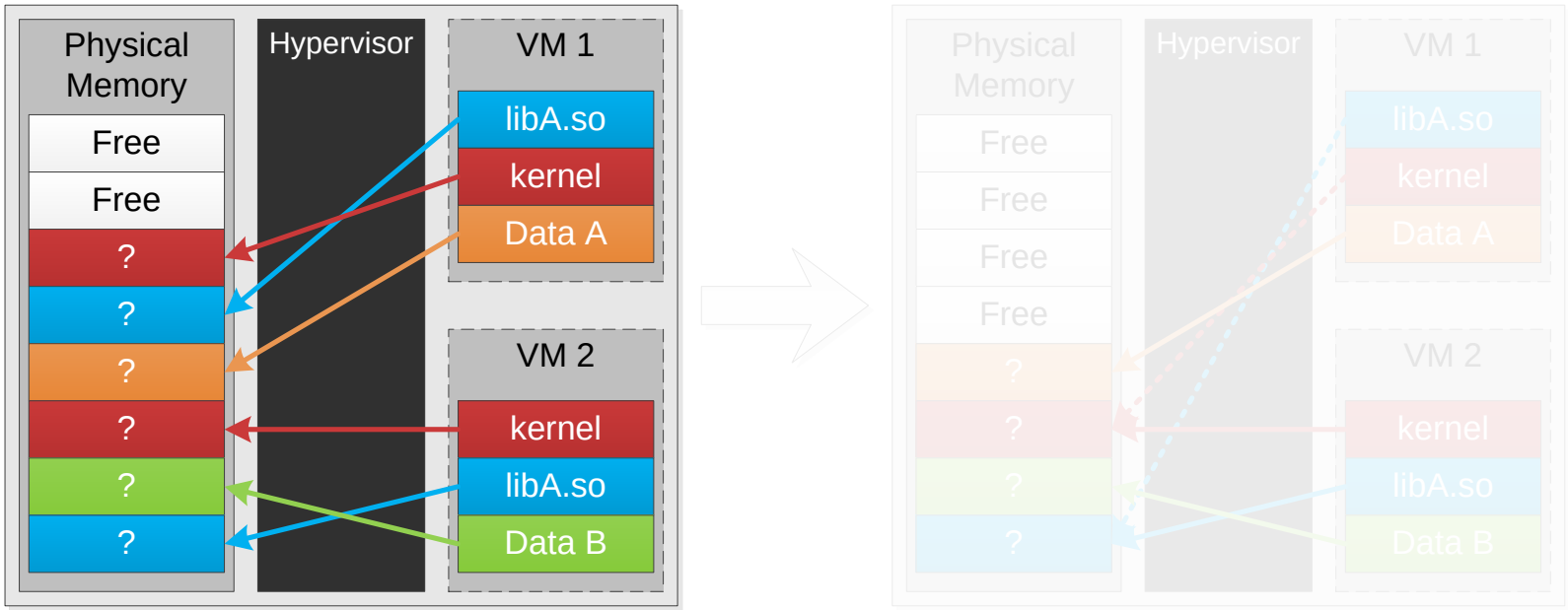
Memory Duplication in VM Environments

- Main memory is the primary bottleneck when consolidating machines
- Different VMs often contain pages with equal content

System	Configuration	Equal pages
VMware ESX (VMware@OSDI'02)	10 VMs, SPEC95	65 %
Difference Engine (UCSD@OSDI'08)	3 VMs, XP/Linux, RUBiS/LAMP	40 % – 85 %
Satori (Cambridge@USENIX ATC'09)	2 VMs, Apache	66 %
Satori (Cambridge@USENIX ATC'09)	2 VMs, Kernel build	11 %
Chang et al (Taiwan Univ@ISPA'11)	Hadoop, HOMP (MPI), LAMP	11 % – 86 %
Barker et al (UMASS@USENIX ATC'12)	offline comparison of desktop/server snapshots	15 %

- Goal: Merge pages, free memory for additional VMs

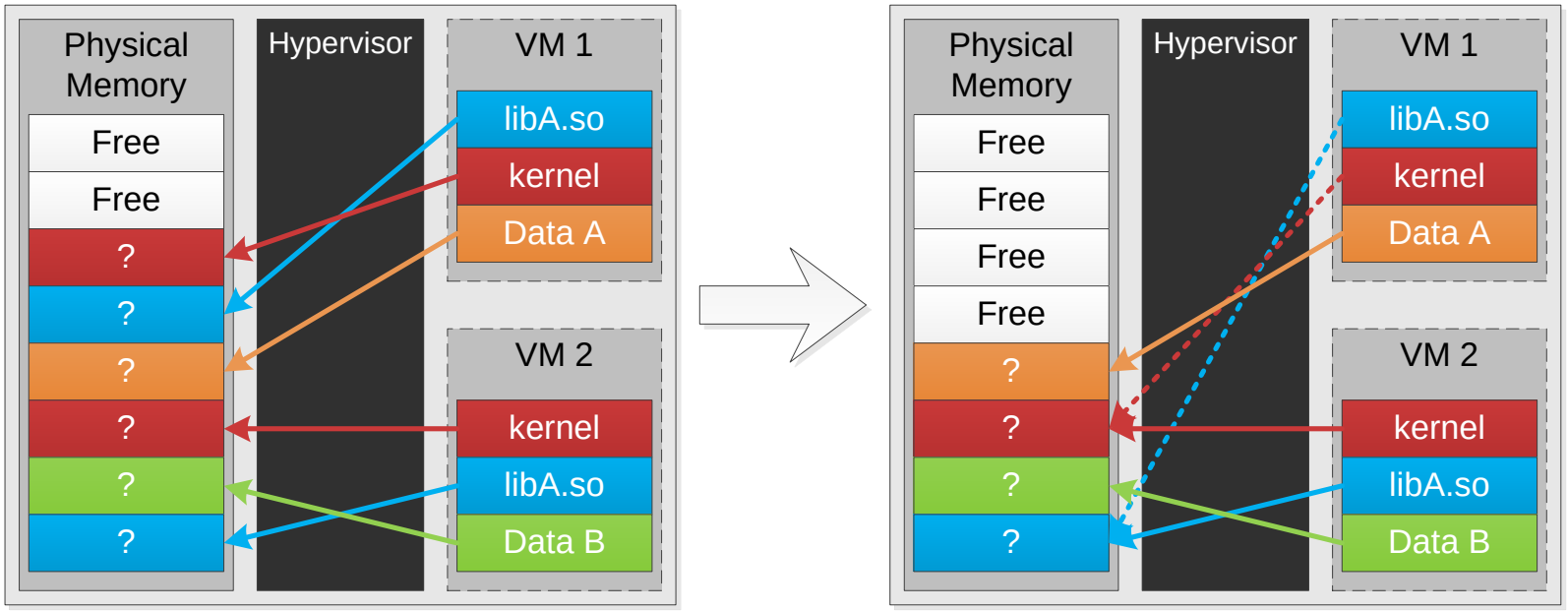
Memory Deduplication for VMs



- *Without deduplication*:
  Every guest page maps to a different host page
- *With deduplication*:
  Pages with identical content are merged and shared between VMs through copy-on-write (COW)

How can pages with equal content be identified?

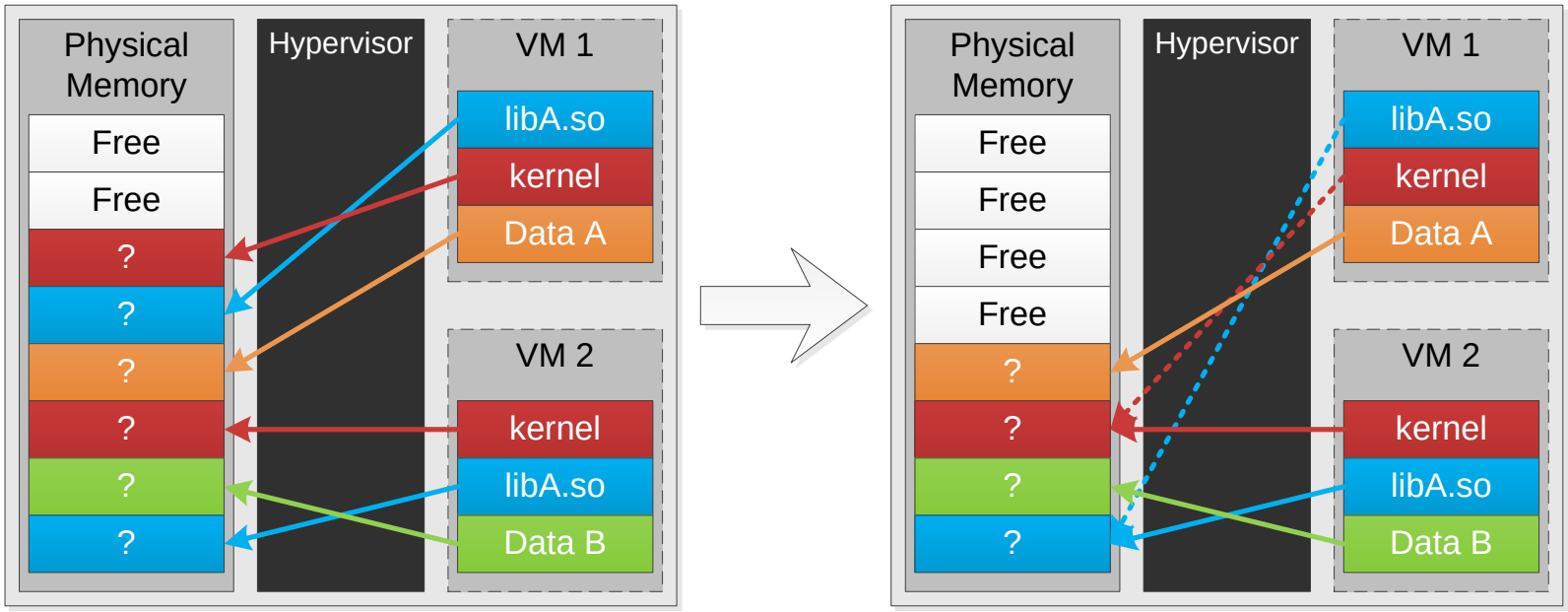
Memory Deduplication for VMs



- *Without deduplication*:
  Every guest page maps to a different host page
- *With deduplication*:
  Pages with identical content are merged and shared between VMs through copy-on-write (COW)

How can pages with equal content be identified?

Memory Deduplication for VMs

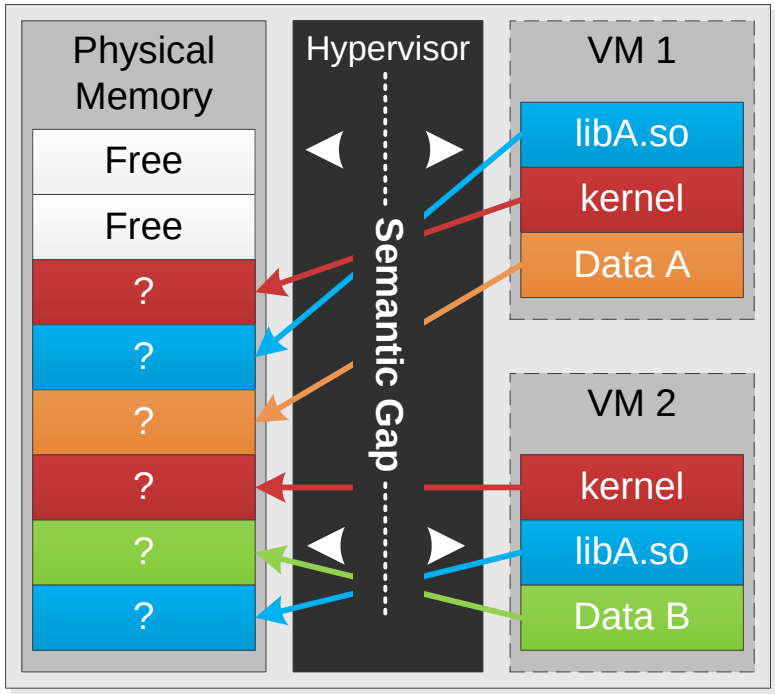


- *Without deduplication*:
  Every guest page maps to a different host page
- *With deduplication*:
  Pages with identical content are merged and shared between VMs through copy-on-write (COW)

How can pages with equal content be identified?

Semantic Gap

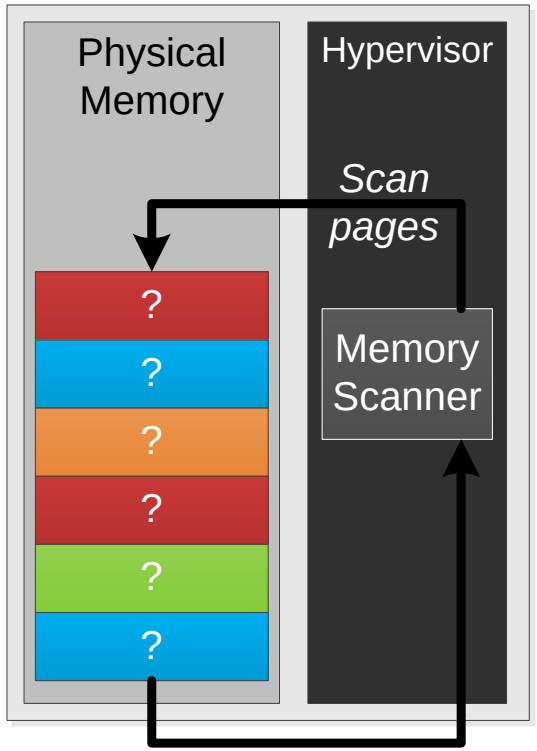
- *Traditional Sharing Mechanisms*:
  Based on source object, not on content
  - `fork()`: parent process
  - `mmap()`: equal inode
- Virtualization introduces *semantic gap* between guest and host
  - Source objects unknown to the host
  - No semantic information about guest pages



Traditional sharing mechanisms cannot be used for deduplicating VMs

Getting Around the Semantic Gap

- **Memory scanners** directly address page content
- Continuously catalog page content
  - Random order (VMware ESX, OSDI'02)
  - Linear order (Linux' KSM, Linux Symposium'09)
- Classify pages based on their modification frequency
  - "Has the page's content changed since last visit?"
- Build index of infrequently modified pages
- Merge/mark COW equal pages that have been found through the index

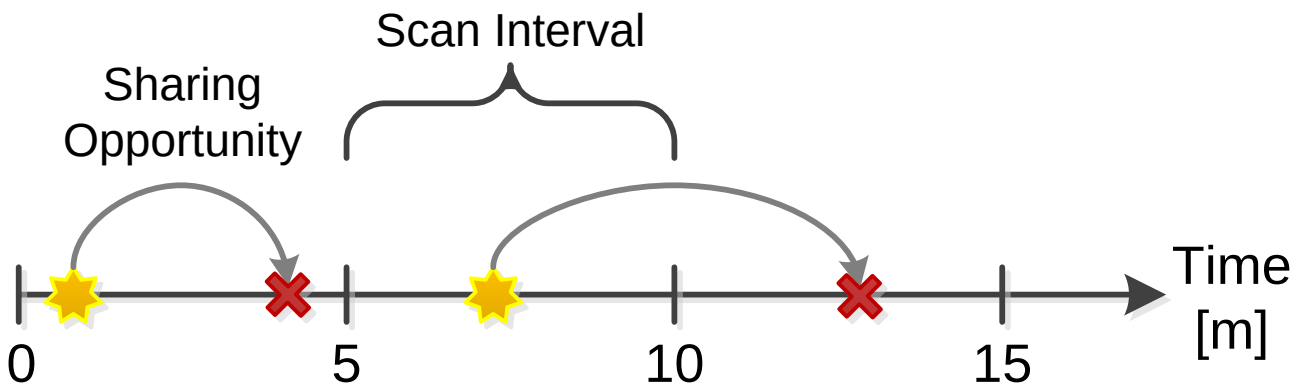


Memory Scanners

- Pay memory density with CPU/memory bandwidth overhead

	Scan Rate	Scan Time	CPU Overhead
Default	1000 $\frac{\text{pages}}{\text{second}}$	5 $\frac{\text{minutes}}{\text{gigabyte}}$	$\sim$ 28 %
Aggressive	5000 $\frac{\text{pages}}{\text{second}}$	1 $\frac{\text{minute}}{\text{gigabyte}}$	$\sim$ 70 %

- Initial benchmarks: more than 70 % of mergable pages modified...
  - ...within a single scan round $\rightarrow$ not caught by scanner
  - ...late enough to amortize the merge cost

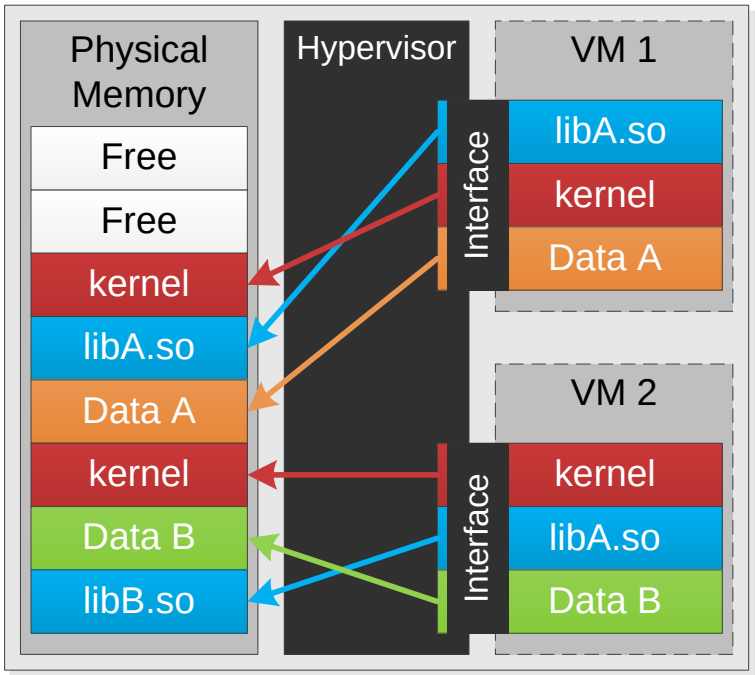


Closing the Semantic Gap

- **Paravirtualization**/Introspection closes the semantic gap

- *Assumption:*
  Many deduplication candidates...
  - ...stem from Virtual Disk Image (VDI) (programs, libraries, data)
  - ...are copies from other data in the system

- Transport information about duplication from guests to host
  - Modify guests' VDI driver (Satori, USENIX'09)
  - Hook guests' syscalls (Disco, SOSP'97)

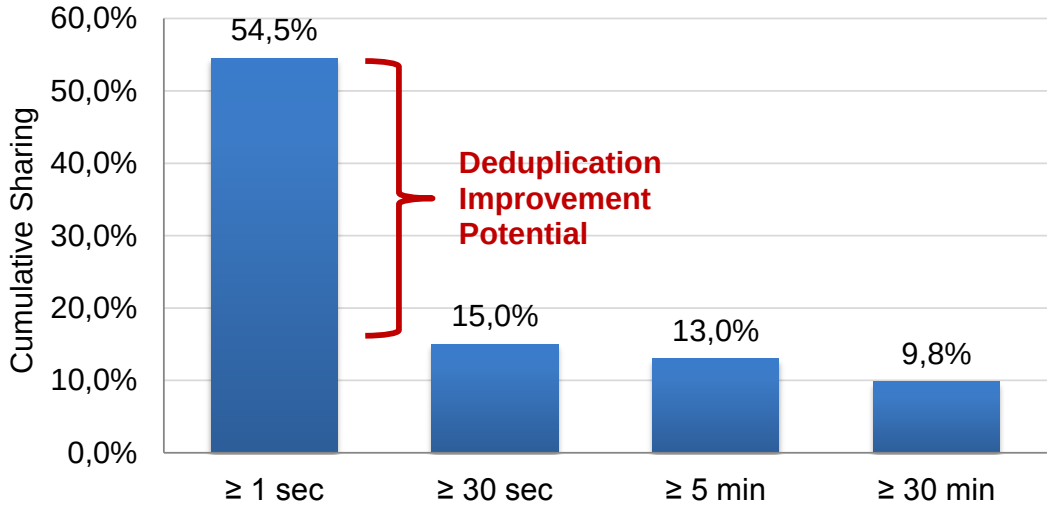


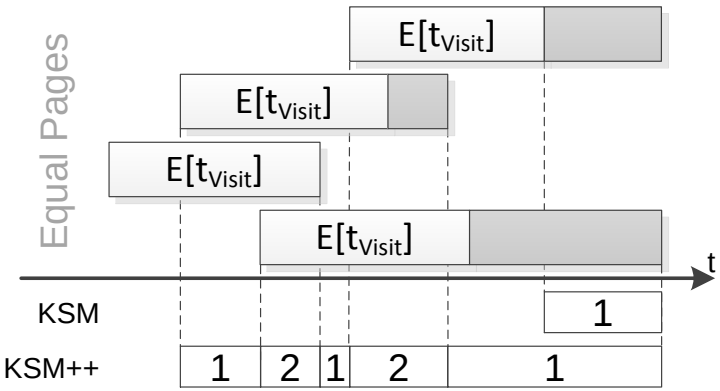
State of the Art

- Memory scanners:
  - Deduplicate sharing opportunities of **any source**
  - Can catch sharing opportunities if they **live long enough** ($>$ 5 – 30 min)
- Paravirtualization based approaches:
  - Deduplicate **short and long-lived** opportunities that stem **from disk**
  - Process **all I/O** $\rightarrow$ Bottleneck for I/O-intensive workloads
- Take-away message:
  - Memory scanners exploit sharing opportunities from all sources
  - Deduplication schemes can be improved through semantic information
  - Guests' I/O pages are prime deduplication candidates

Temporal Memory Duplication Characteristics

- 3 VMs: Ubuntu + Firefox + {LibreOffice, Gimp, Eclipse} in Simics





- Sharing opportunities live. . .
  - . . . extremely short → not worth sharing
  - . . . between 1 sec – 30 sec → not caught by memory scanners
  - . . . long → already caught by memory scanners

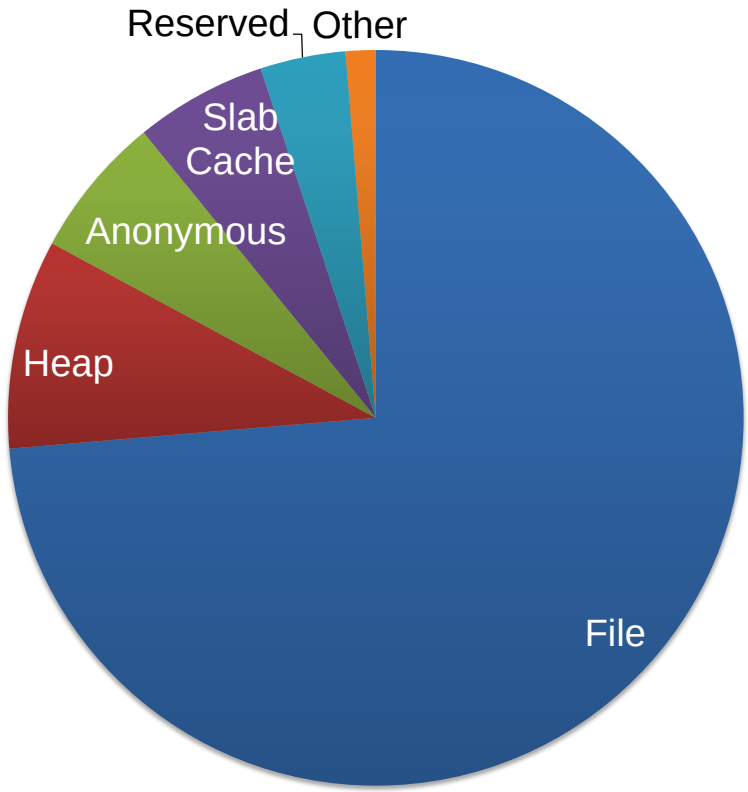
Visiting sharing opportunities earlier leads to more deduplicated pages

Semantic Memory Duplication Characteristics

- 3 VMs: Ubuntu + Firefox + {LibreOffice, Gimp, Eclipse} in Simics

Memory Category	Prop. of Sharing
File	73.7 %
Heap	9.2 %
Anonymous	6.3 %
Slab Cache	5.8 %
Reserved[1]	3.8 %
Other	1.3 %

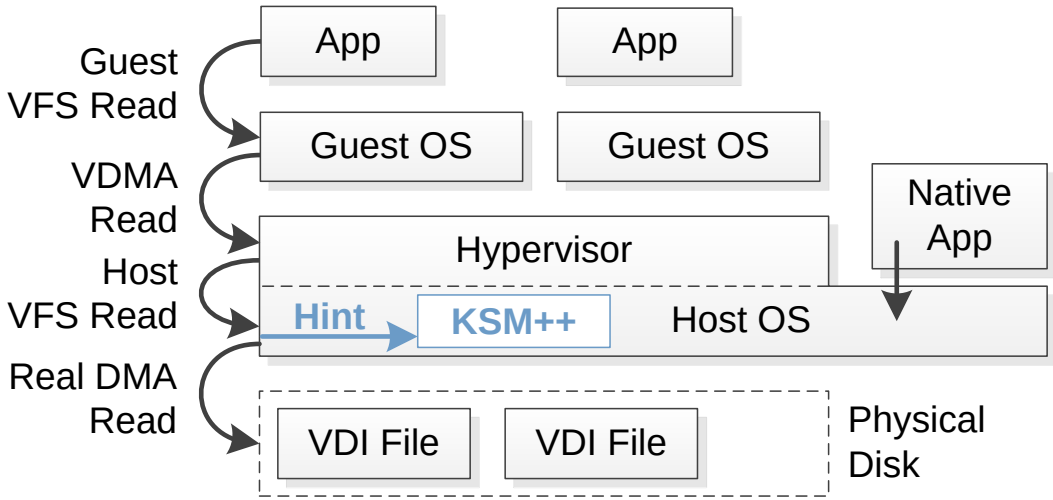
[1] Non-free pages not explicitly tracked by OS introspection (e.g., driver private pages)



- Barker et al.: 50 % Heap, 43 % File
- Kloster et al.: 64 % – 94 % File

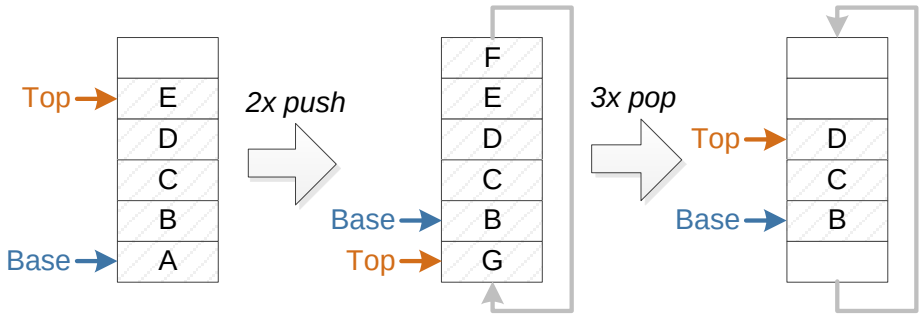
KSM++: Hints for Memory Scanners

- Best of both worlds: Integrate I/O-based dedup into memory scanner
- Host/Hypervisor does I/O on behalf of guest VMs
  - I/O-operations target guests' buffer caches and mmap areas
  - Record Host-VFS target memory areas in a "Hints Buffer"
- Visit I/O-pages earlier in memory scanner
- No paravirtualization required
  - guest-agnostic
  - also works for native apps (e.g., Zero Install)



Storing and Processing Hints

- Hints are buffered in a *bounded circular stack*
  - Keeps history of last unprocessed `$stack_size` disk accesses
  - Bounded memory requirements, e.g., during I/O-burts
  - Implicit pruning and aging

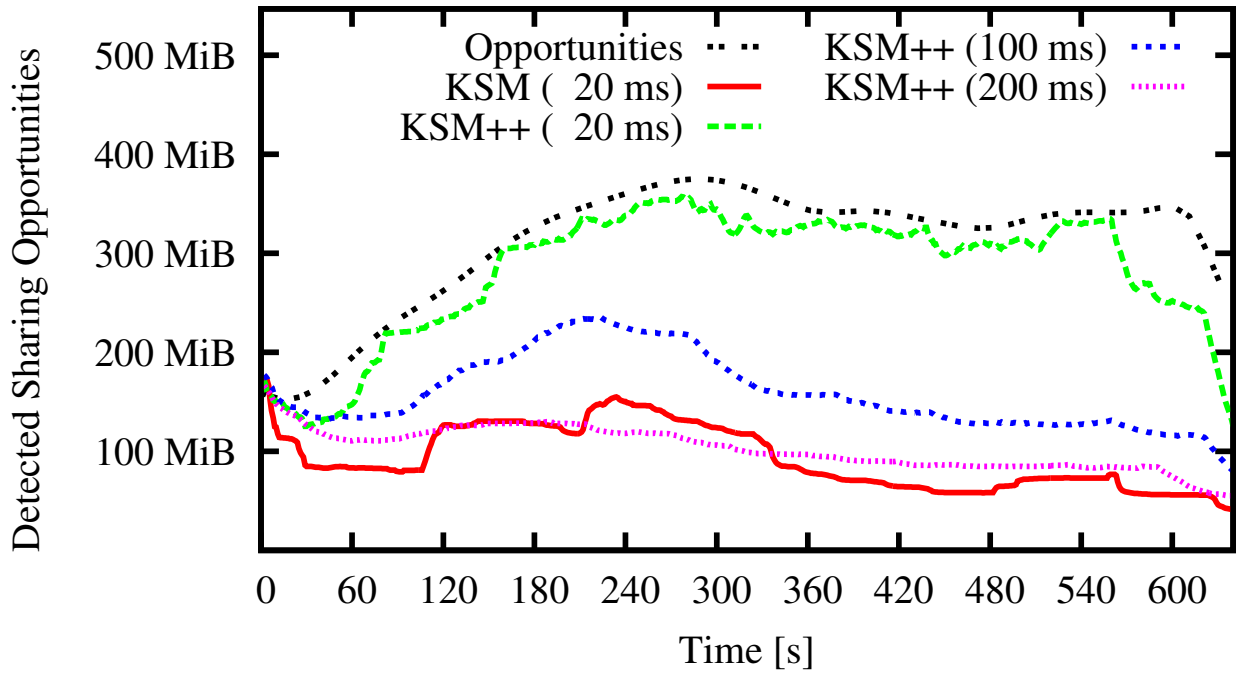


- KSM daemon loops through all virtual mappings
  - Wakes up periodically and scans a fixed number of pages

- KSM++ decides on wakeup if scanning or processing hints
  - Processes hints **interleaved** to regular KSM scan
  - Does not starve non-I/O scan $\rightarrow$ catches duplicates from all sources
  - Obeys scan rate limits (can limit CPU/IO resource consumption)

Merge Performance: Kernel Build

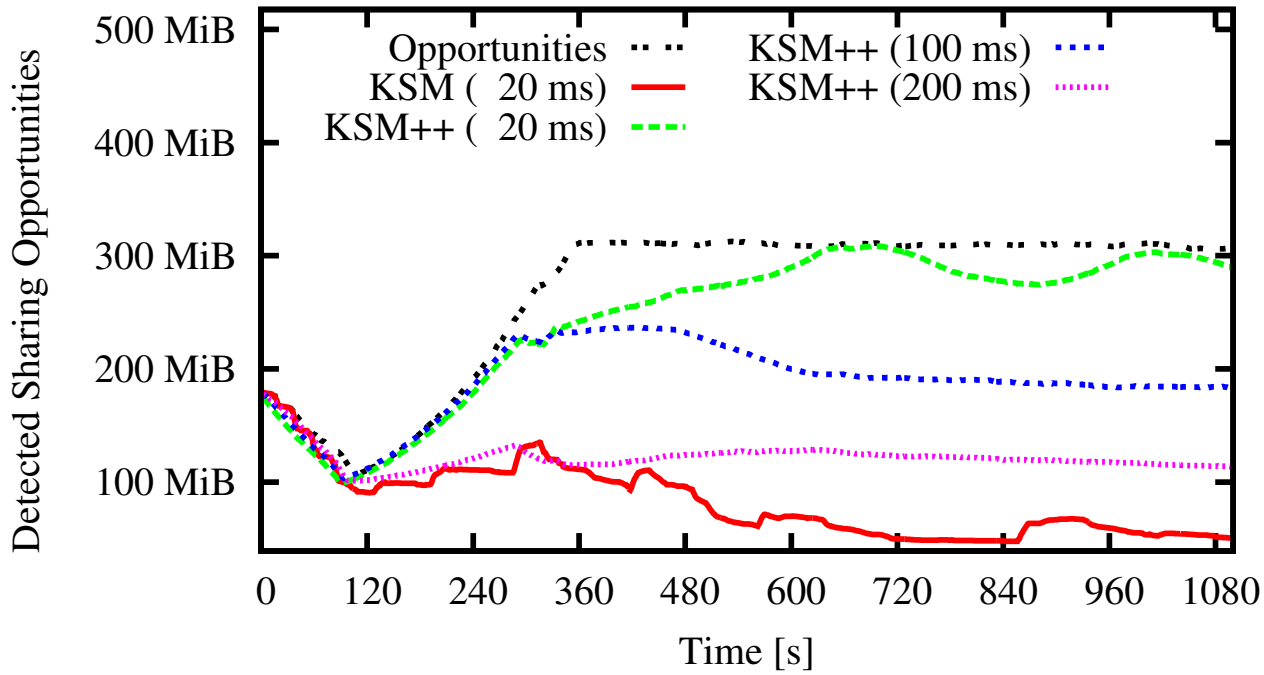
- 2 VMs: Linux kernel build
  - Default scan rate: 5000 $\frac{\text{pages}}{\text{second}} \rightarrow$ 100 pages every 20 ms



- Opportunities peak at about 37 % of total memory assigned to both
  - Opportunities determined with 1s snapshots
- Measured same benchmark runtimes for KSM and KSM++

Merge Performance: Apache + HTTPerf

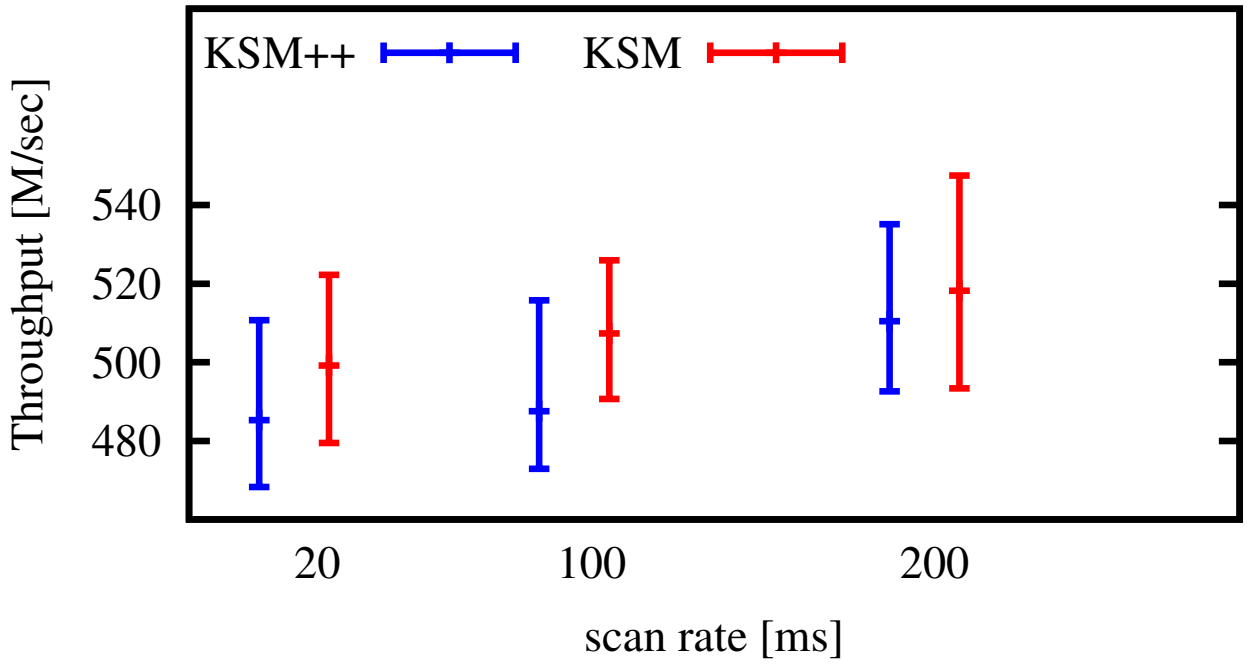
- 2 VMs: Apache, serving the same set of files
  - Sum of served files does not fit into main memory
  - Different, random access order for both VMs



- Higher line = more pages shared = more memory saved
- Measured same throughput with HTTPerf

Overhead of Hint Generation

- 1 VM: Bonnie++ stress test
  - Average of 30 measurements with .05 and .95 quantiles



- Disk throughput does not vary significantly when choosing KSM++

KSM++ Overhead

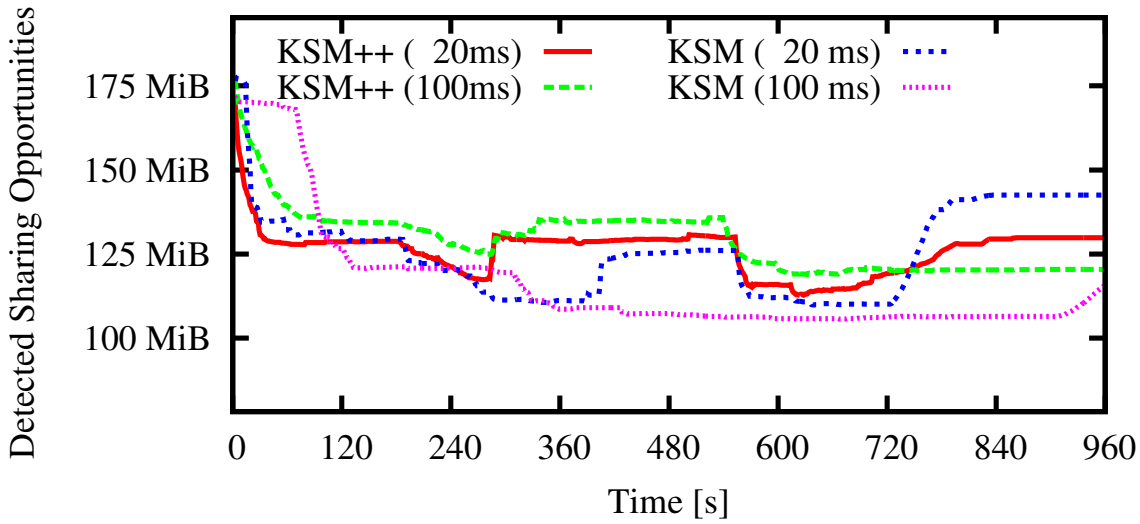
- CPU consumption:

Approach	20 ms	100 ms	200 ms
KSM	68.8 %	27.5 %	16.3 %
KSM++	67.1 %	33.6 %	17.0 %

- Negligible additional memory consumption
  - Hint buffer $\rightarrow$ 2 MiB
  - Lock for serialization of buffer accesses
- Runtime variation between KSM and KSM++ below 1 %
- Breaking shared pages may happen at a bad time
  - `malloc` $\rightarrow$ initialize with pattern $\rightarrow$ deduplicate $\rightarrow$ write
  - This is why we don't merge the free-pool (zero-pages)
  - Not due to hinting but due to more effective deduplication

Worse deduplication through hints?

- Nothing to share? → can't get worse/no difference
- No I/O? → no hints → scan rate is fully used for linear scan
- Worst case: Many sharing opportunities **not** based on files
  - Hints slow down detection of sharing opportunities
  - Interleaving ratio limits how much worse it gets
  - e.g., 1:1 → memory scan at most twice as slow
- Mixed workload (1. VM: Apache, 2. VM: Kernel build):



Future Work I

- Enable/Disable I/O-hints based on static analysis of used VDI's
  - Turn off hinting if VDI's are very different
- Dynamically adapt settings
  - Scan rate: based on merge success
  - Interleaving ratio: based on merge success of hints/scan
  - Buffer size: based on scan rate and page fluctuation

Future Work I

- Enable/Disable I/O-hints based on static analysis of used VDI's
  - Turn off hinting if VDI's are very different
- Dynamically adapt settings
  - Scan rate: based on merge success
  - Interleaving ratio: based on merge success of hints/scan
  - Buffer size: based on scan rate and page fluctuation

Future Work II

- Incorporate hints from other sources
  - TLB-miss handler
- Statistical analysis of sharing history via full system simulation
  - Which page histories predict sharing opportunities?
  - Which pages are overwritten with same content?
- NUMA-aware memory deduplication
  - Remote memory accesses are expensive: + 75 % latency, - 33% bandwidth
    - Worst case: all pages on remote node (e.g., SPEC libquantum: 2 $\times$ run time)
    - High page access frequency $\rightarrow$ avoid sharing across nodes
    - Which nodes reference a certain page?
    - Revoke deduplication, replicate shared pages
  - Storage class memory (PCM, STT-RAM) shows poor write characteristics
    - Deduplicated pages are good candidates for SCM due to long-lasting RO/COW mapping

Future Work II

- Incorporate hints from other sources
  - TLB-miss handler
- Statistical analysis of sharing history via full system simulation
  - Which page histories predict sharing opportunities?
  - Which pages are overwritten with same content?
- NUMA-aware memory deduplication
  - Remote memory accesses are expensive: + 75 % latency, - 33% bandwidth
    - Worst case: all pages on remote node (e.g., SPEC libquantum: 2 $\times$ run time)
    - High page access frequency $\rightarrow$ avoid sharing across nodes
    - Which nodes reference a certain page?
    - Revoke deduplication, replicate shared pages
  - Storage class memory (PCM, STT-RAM) shows poor write characteristics
    - Deduplicated pages are good candidates for SCM due to long-lasting RO/COW mapping

Future Work II

- Incorporate hints from other sources
  - TLB-miss handler
- Statistical analysis of sharing history via full system simulation
  - Which page histories predict sharing opportunities?
  - Which pages are overwritten with same content?
- NUMA-aware memory deduplication
  - Remote memory accesses are expensive: +75 % latency, -33% bandwidth
    - Worst case: all pages on remote node (e.g., SPEC libquantum: 2 $\times$ run time)
    - High page access frequency $\rightarrow$ avoid sharing across nodes
    - Which nodes reference a certain page?
    - Revoke deduplication, replicate shared pages
  - Storage class memory (PCM, STT-RAM) shows poor write characteristics
    - Deduplicated pages are good candidates for SCM due to long-lasting RO/COW mapping

Conclusion

- Main memory is scarce in virtualized environments $\rightarrow$ deduplication
  - Memory scanners can find long-lived sharing opportunities
  - I/O-based systems can find short lived opportunities

- KSM++: Combination of memory scanning and I/O-based approaches
  - Deduplicate pages from all sources (named **and** anonymous)
  - Quick detection of VDI-based sharing opportunities
  - Lossy buffer copes with bursty I/O
  - Configurable, limited overhead
  - No paravirtualization

- KSM++ hints may help detecting up to 4x more sharing opportunities than pure random or linear scanning in our benchmarks