Improving Virtualized Storage Performance with Sky

VEE '17

Leo Arulraj,
Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau
ADSL Research Group, UW-Madison

Poor Virtualized Storage Performance

Virtualization ubiquitous in data centers, cloud services

VMs and Virtualized Storage have many advantages

- consolidation
- cost reduction
- high availability
- better security
- backups
- snapshots

Problem: Poor performance compared to native storage

Due to disk emulation, host storage stack involvement etc.

Challenging to achieve good virtualized I/O performance

Rosenblum and Waldspurger *ACM Queue Virtualization '11*

Virtualized I/O is 35% to 45% slower than native I/O

Sorensen *LinuxCon '11*, Shafer *WIOV '10*

Information is key for managing resources

Hypervisor is the best place to address this

Managing resources effectively requires information

Better I/O scheduling for improved throughput and latency:

- Guest task I/O priority [Kim et al. *EuroPar '08, VEE '09*]
- Guest task scheduling details for anticipatory scheduling [Antfarm: Jones et al. *Usenix '06*]

Better memory management:

- Which blocks are cached by guest VM and hence need not be cached by VMM? [Geiger: Jones et al. *ASPLOS '06*]
- Working-set size to balance memory [MEB: Zhao et al. *VEE '09*]

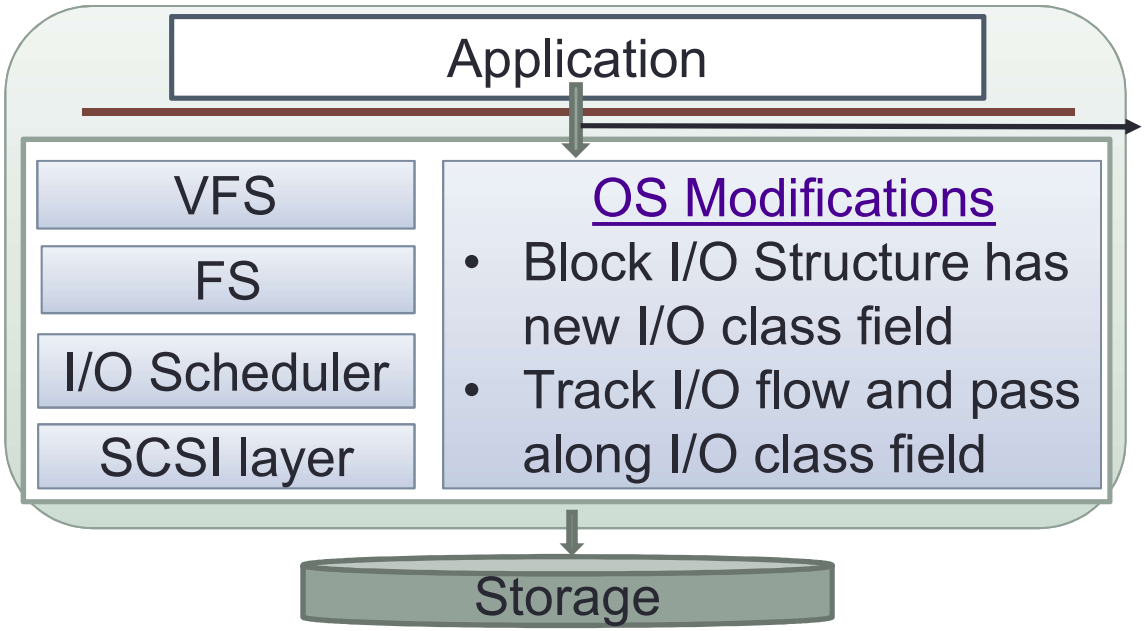
Better VM consolidation:

- VM's resource usage for migration [Sandpiper: *NSDI '07*]

How to obtain information about I/O?

One approach: Modify storage layers and interfaces to pass semantic information along with I/O requests

Differentiated Storage Services [Mesnier et al. *SOSP '11*]
Modified Linux/Ext3 (using kernel patches) and Windows/NTFS (using filter drivers)



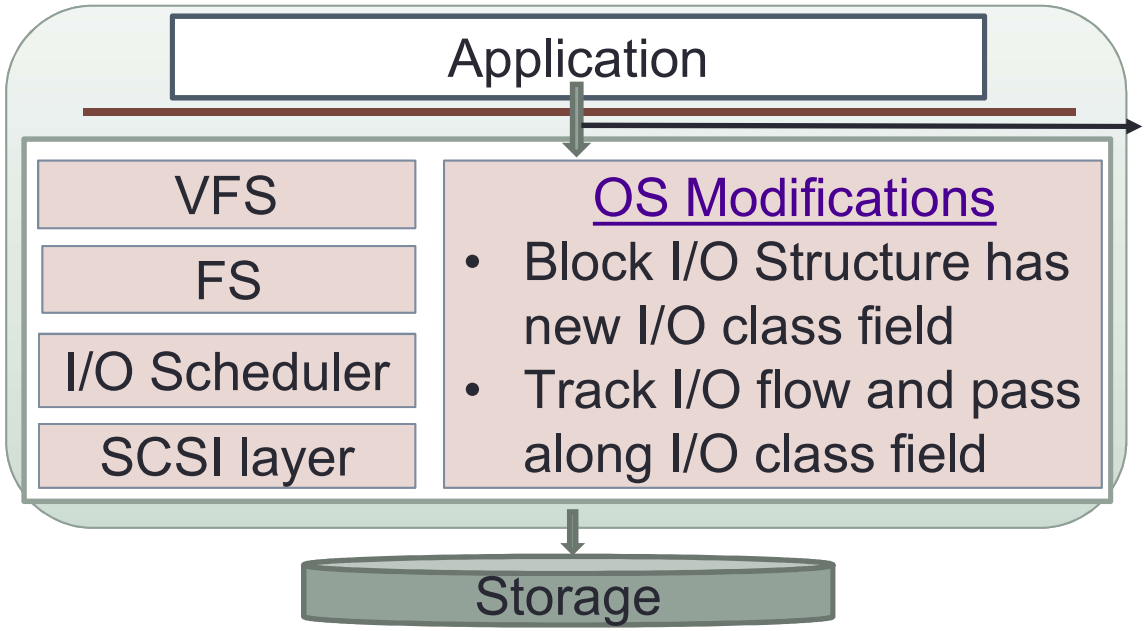
readv,writev system-calls take extra I/O-class argument

OS, VFS, FS, I/O Scheduler, SCSI layer all need changes

Problems with modifying interfaces

Hard to deploy, maintain and adopt because of changes across the storage stack

Have to redo similar changes again to support new FS or OS (could consume lot of time and effort)



Support new FS or OS?
Modify again!

Cross-layer changes not always needed!

An implicit approach to gathering information in VMM

Intercept system-calls made by guest applications

Advantages of this implicit approach:

- Easy to deploy because no changes to FS or OS
- Portable across new OSes due to similarity of system-call interface because of POSIX standard

Sky uses this implicit approach to gather information and insights about guest I/O requests

Overview of Sky

Sky uses system-call interception to:

- passively monitor system-calls
- inject new system-calls
- allow applications to supply I/O hints with system-calls

Sky uses these techniques to gain semantic information about guest I/O requests

Sky improves virtualized-storage performance with better caching and deduplication systems

Three case studies using Sky

Information gathering:

- Amount of guest FS metadata, block lifetimes

iCache: Smart Caching (2.3x to 8.8x improvement)

- Higher priority to guest FS metadata and small files
- MySQL clustered index over secondary index

iDedup: Smart Dedup (4.5x to 18.7x improvement)

- Don’t dedup encrypted data
- Cache disk-backed mappings for file-copy workload

Talk Outline

- ✓ Motivate Sky
- ➢ Generic techniques for system-call interception

I/O semantic information gathering (Insights)

Prototype

Case studies and evaluation

Conclusion

Techniques

Sky uses three main techniques:

- passively monitors guest application's system-calls
- injects system-calls to read guest-OS state
- allows applications to supply I/O hints with system-calls

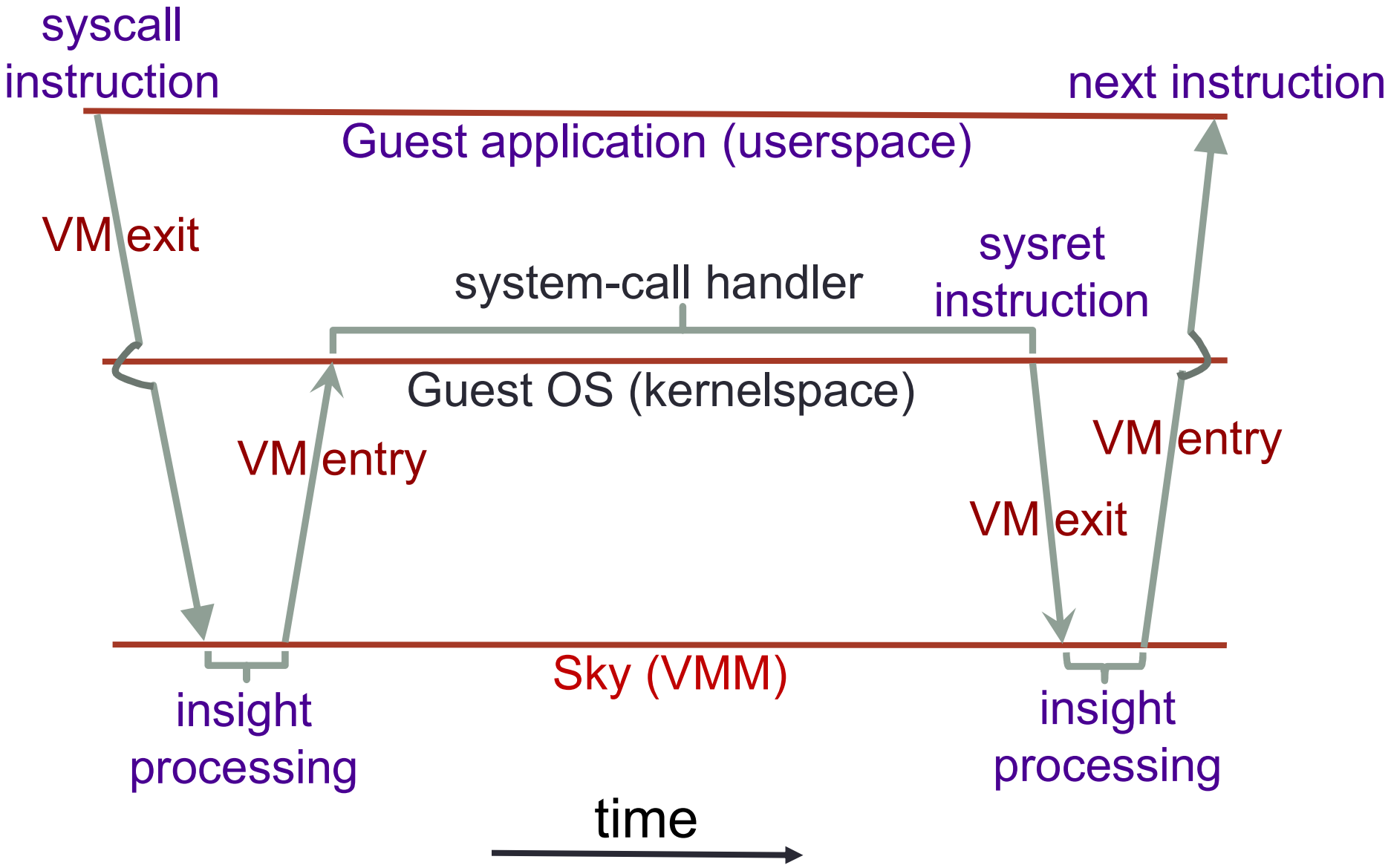
How to efficiently gather accurate information?

- Hardware system-call interception mechanism
- Selectively monitor only a targeted set of processes
- Handle special cases:
  - Signal handler invocation
  - Context switches in system-call handler
  - Misaligned I/O, Small I/O, memory-mapped I/O etc.

System-call interception

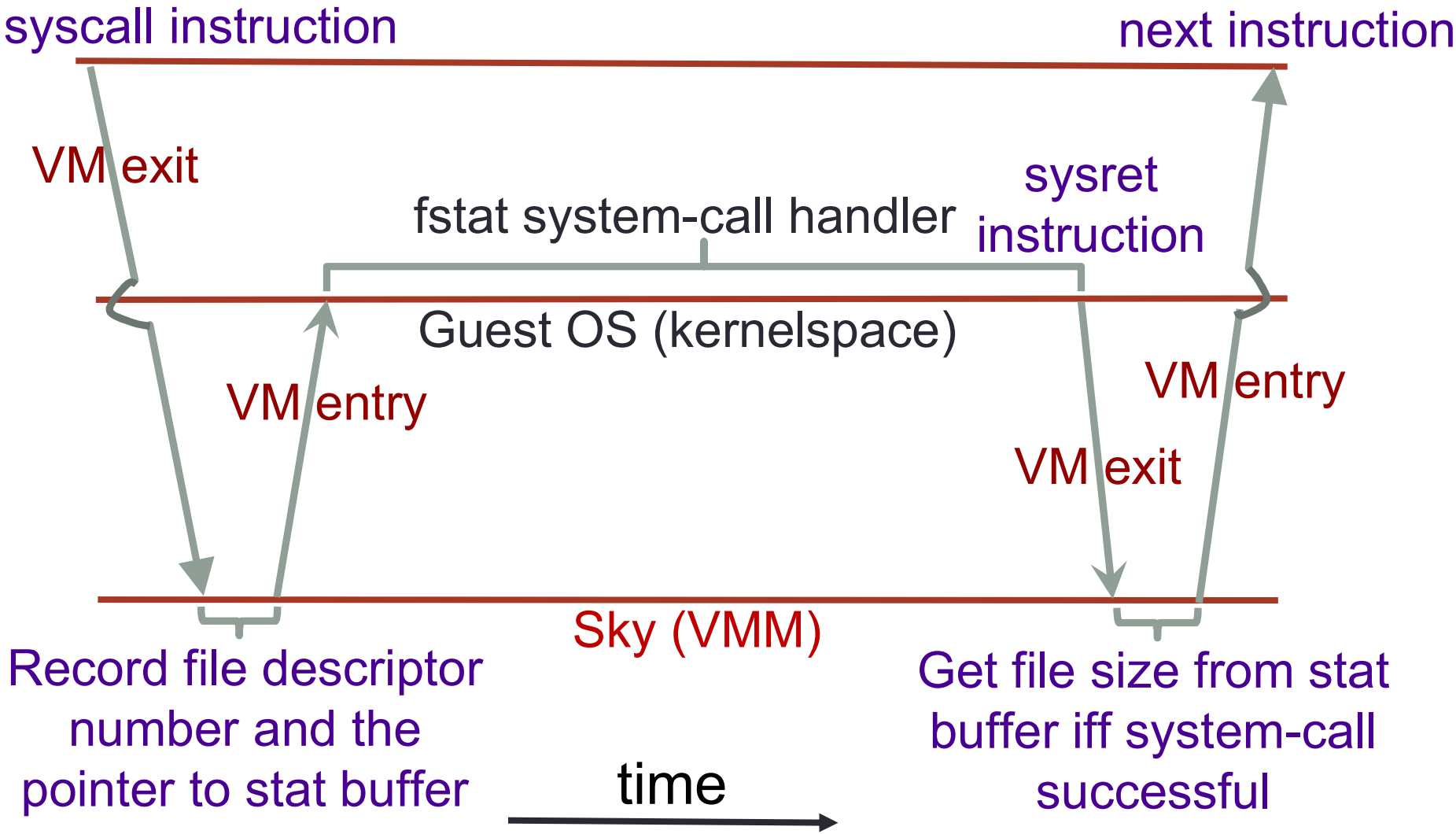
Use hardware controls to intercept system call and return instructions (e.g. SYSCALL/SYSRET: unsetting SCE flag faults when executing system calls)

System-call interception



How to track File Size?

[...assume application calls *int fstat(int fd, struct stat* buffer)*...]



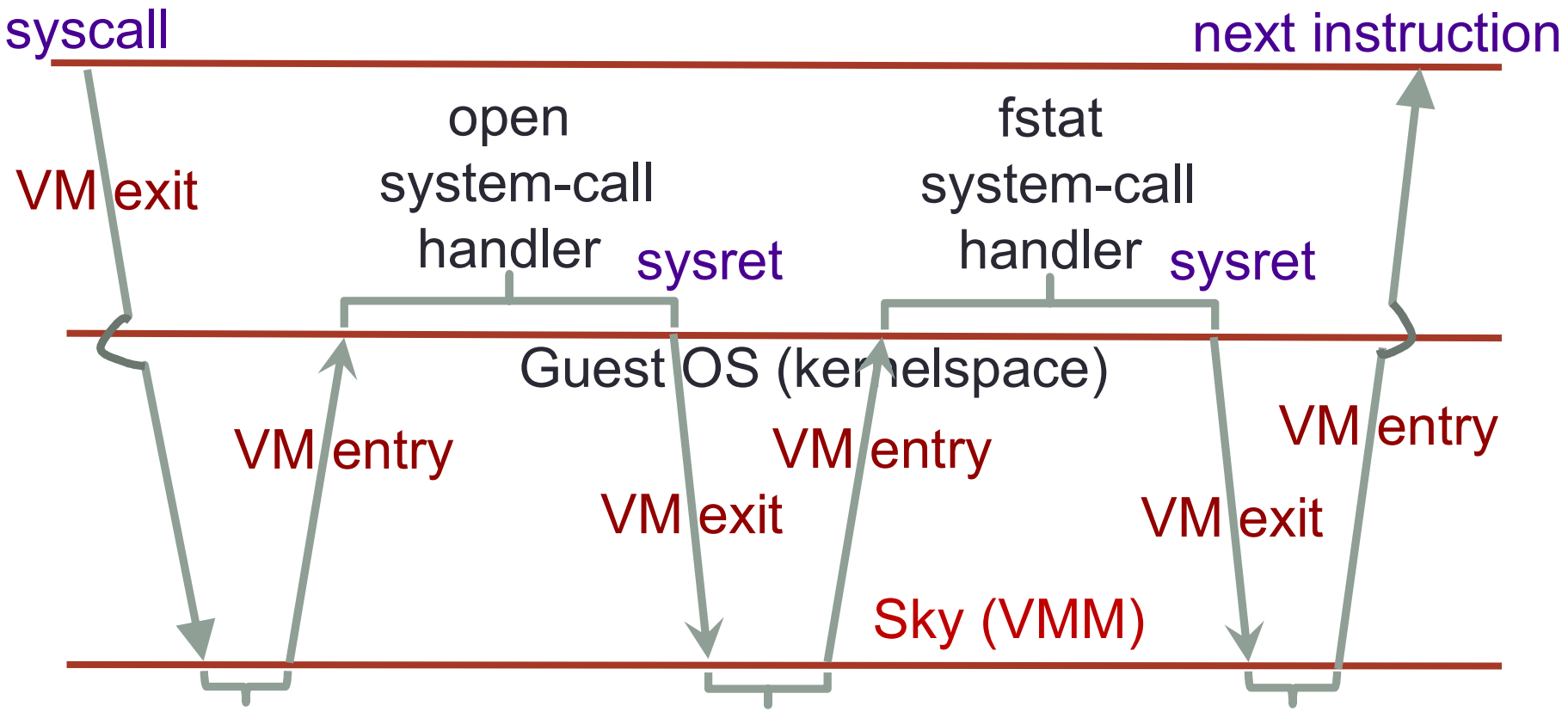
Insight-Calls

Question: What if application does not call fstat on a file descriptor?

Answer: Sky issues *fstat* insight-call during file open and then tracks I/O related system-calls like read, write, seek, etc.

Insight-Calls

[...assume application calls *int open(char* path, int flags)*...]



1. Record file path
2. Record file descriptor
3. Write registers with fstat arguments
4. Reset IP to syscall instruction
5. Get file size from stat buffer
6. Replace return value with file descriptor

Insight-Calls

Insight-Calls are:

- Issued in context of monitored process
- Used only to read guest-OS state

Used in many scenarios by Sky:

- Get process identifier (PID) of monitored process
- Handle misaligned I/O by reading partial file contents to calculate 4 KB aligned checksums
- Get current working directory when dealing with relative file paths

Application Supplied Insights

Allowing supplied I/O hints brings good benefits

For example:

- Cloud file server gives higher priority to premium customers
- Database server gives higher priority to clustered index

Mechanism: Pass a magic number and I/O class as last two additional arguments of a system-call

Also have convenient wrappers like:

iwrite(file descriptor, buffer, size, ioclass)

instead of

write(file descriptor, buffer, size)

Selective Process Interception

Question: Should Sky monitor all system-calls issued by all processes in the guest OS?

Answer: No. Sky only needs to intercept a targeted set of processes.

Avoids unnecessary overheads.

Selective Process Interception

Enable hardware interception only for monitored processes

Automatically monitor child processes by intercepting process management related system-calls like fork, clone, exit, kill etc.

Tracking Guest OS Scheduling: CR3 register overwrites are intercepted

Process Identification:

- CR3 register contains Page Directory Base Address
- GetPid() insight-call to get the guest OS assigned PID

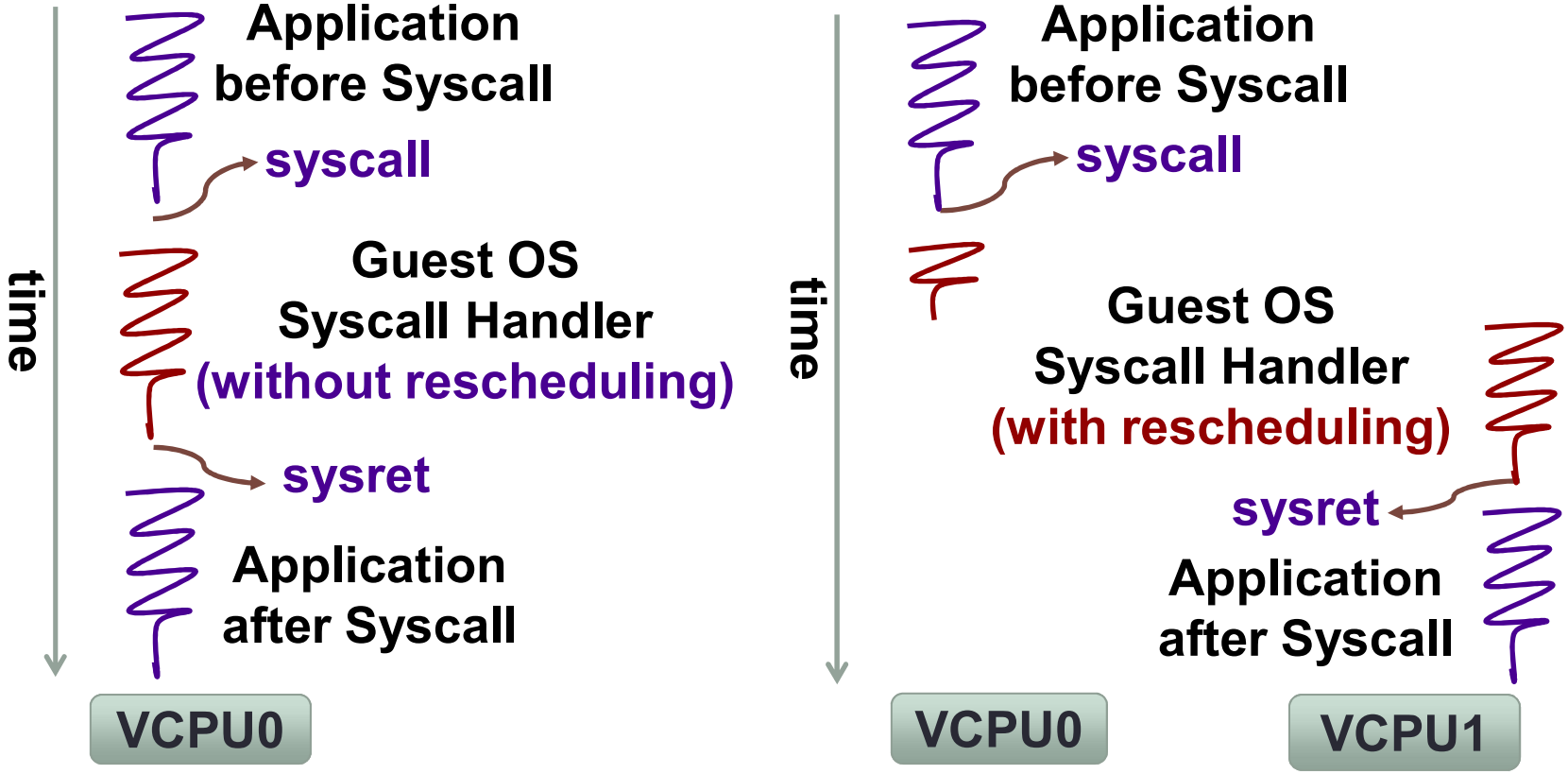
Thread Identification: Use Stack Pointer register contents

Handling Process Rescheduling

Question: How to handle tricky scenarios like context-switch while executing system-call handler?

Syscall entry and exit split across different virtual processors

Handling Process Rescheduling



Case 1: **No** Rescheduling

Case 2: **With** Rescheduling

Application

Kernel

Matching Syscall Exit with its Entry

Upon context-switch, store outstanding system-call entry details into global hashtable with PID as key

Upon an unmatched system-call exit

1. Match with the stored system-calls
2. Match with the outstanding system-call entry in other processors (no new process yet scheduled on previous processor)

Signal Handler invocations are also handled similarly (details in paper)

Talk Outline

- ✓ Motivate Sky
- ✓ Generic techniques for system-call interception
- ➢ I/O semantic information gathering (Insights)
  - Insight #1: File Size Tracking
  - Insight #2: Guest FS metadata vs. data classification
  - Insight #3: Application I/O patterns – encryption, file-copy

Prototype

Case studies and evaluation

Conclusion

Insight #1: File Size Tracking

Track syscalls involving file descriptors:

- Intercept I/O system-calls like open, write etc.
- Track mapping of file descriptors to files
- Track current file descriptor offsets and file sizes

Associate file size information with I/O requests

- iCache gives higher priority to small files
- Because large files usually sequential on disk

Selective monitoring of files (e.g. file path prefix)

Insight #2: Guest Application Data Identification

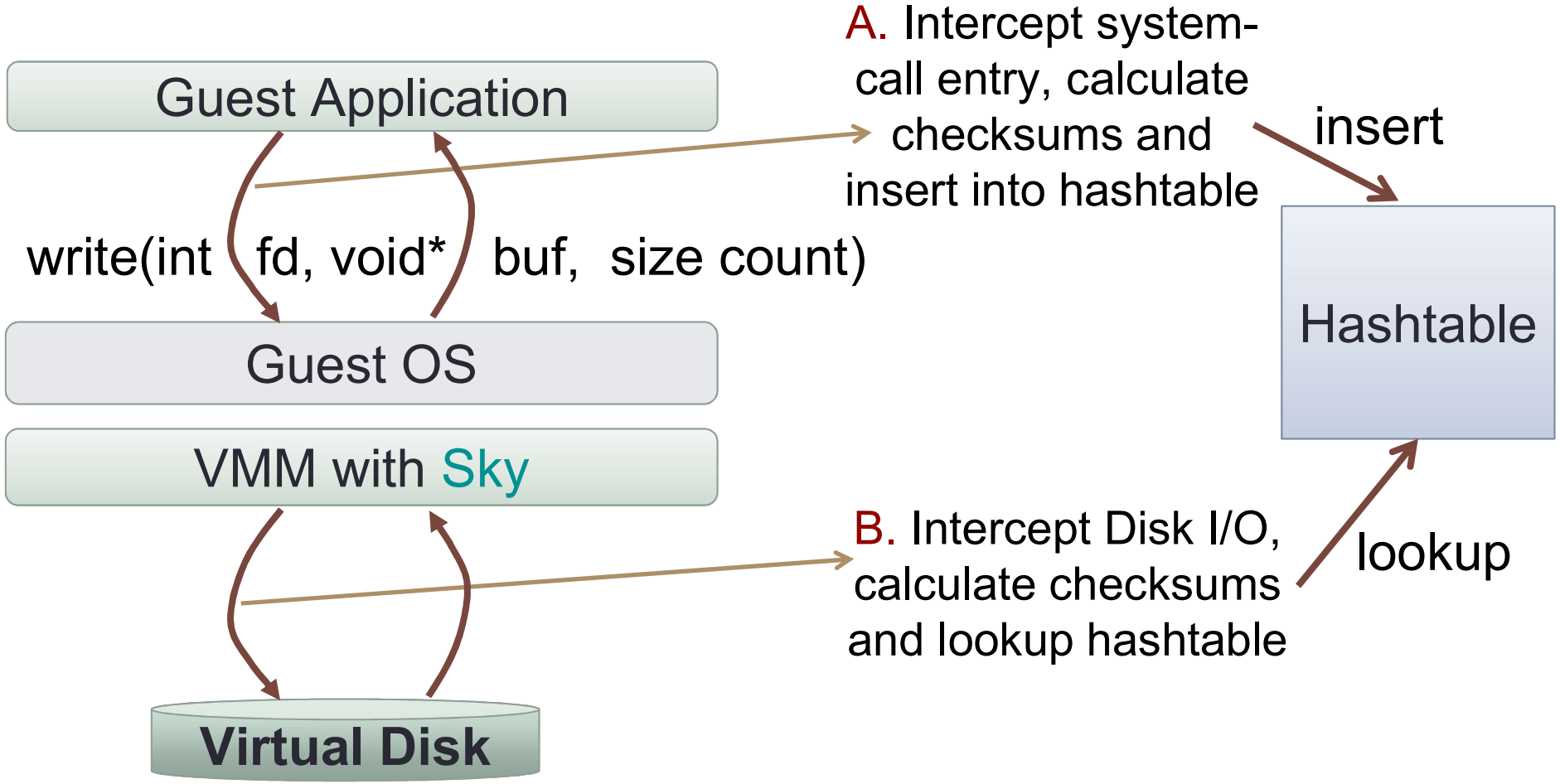
Used by iCache to give higher priority to guest FS metadata over application data

How to identify guest FS metadata from data?

- Guest FS metadata originates from guest OS kernel space
- Guest application data originates from userspace using system-calls

Insight #2: Guest Application Data Identification

Handling Write I/O Requests



Handling Memory-Mapped I/O

Upon initial *mmap* syscall:

- Record virtual address range
- Write protect corresponding guest process page table entries

Tracking page remapping by Guest OS:

- Write protection traps on page remapping
- Track corresponding host machine pages

Identifying Guest Data I/O:

- Intercept all I/O to virtual disk
- Check their buffer addresses against tracked *mmap* pages

Details on handling reads, misaligned and small I/O in paper

Insight #3: Application I/O Patterns

File copy and encryption patterns can be used to improve deduplication performance

Detecting File Copy

- Compare checksums of blocks read and written by a process
- Repeated matches mean file-copy I/O pattern

Detecting File Encryption

- Use names of executables as hint (e.g. gpg)
- Use file name extensions of the destination files as hint

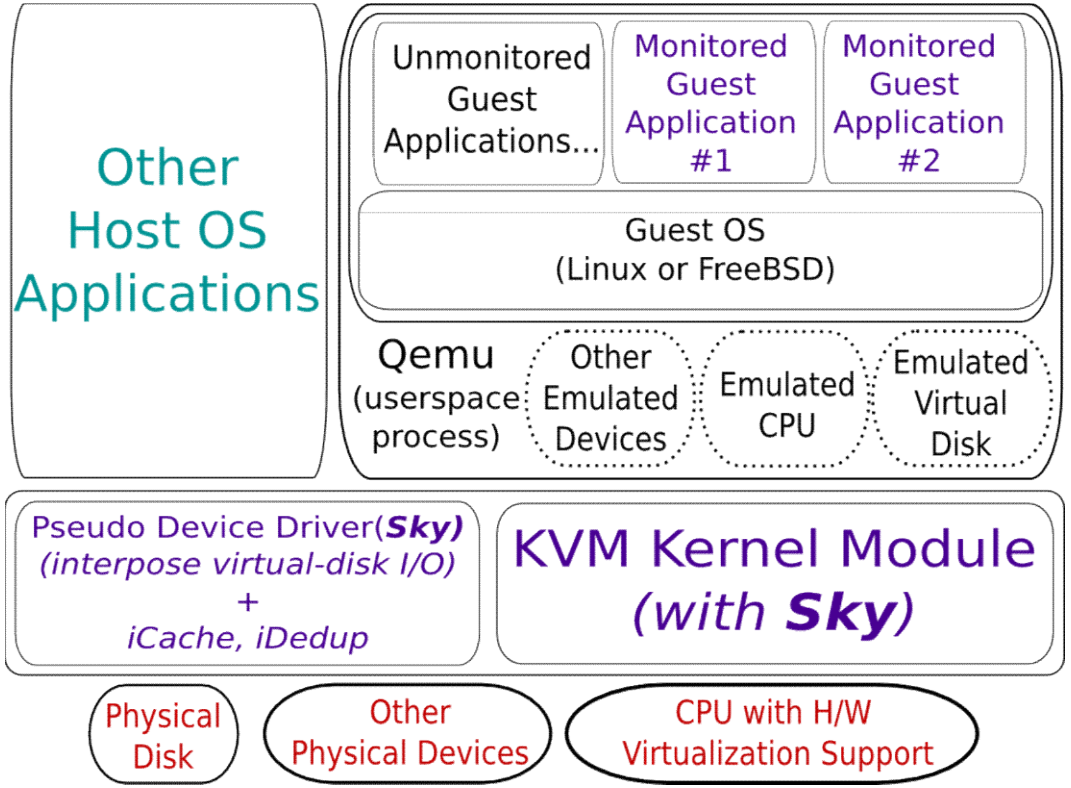
Talk Outline

- ✓ Motivate Sky
- ✓ Generic techniques for system-call interception
- ✓ I/O semantic information gathering (Insights)
- ➢ Prototype
  - Overview
  - Efficiency

Case studies and evaluation

Conclusion

Prototype of Sky



Sky is 7.8 KLOC of new code added to 43.6 KLOC of KVM

Pseudo device driver in the host (3.8 KLOC of source code)

Bcache (10% new code), Dmdedup (14.5% new code)

Prototype Efficiency

Is Sky's system-call interception overhead tolerable for I/O workloads? Yes (under 5%)

Workload	With Sky (secs)	Without Sky (secs)	% Overhead
Encryption	25.3	24.8	2
File Search	54.6	53.4	2
File Copy	34.9	34.7	1
Mail Server	163.5	160.8	2
Database Server	21	21.7	-3

Talk Outline

- ✓ Motivate Sky
- ✓ Generic techniques for system-call interception
- ✓ I/O semantic information gathering (Insights)
- ✓ Prototype
- ➢ Case studies and evaluation
  - Case Study #1: Information Gathering
  - Case Study #2: iCache (smart caching in VMM)
  - Case Study #3: iDedup (smart deduplication in VMM)

Conclusion

Case Study #1: Information Gathering

Guest FS metadata-data classification is accurate:

- Can be used to study behavior of various workloads on different guest FS

Block lifetime information has many potential uses:

- Schedule write-back caching
- On-disk data reorganization
- Intelligent prefetch and fast recovery that skip dead blocks

Sky intercepts *unlink, truncate* etc. to calculate accurate block lifetime information

Results and graphs on information gathering in paper

Case Study #2: iCache

Bcache module with modifications (10% new code):

- Update priority of cache slots based on I/O class from Sky
- Collect and report statistics at sector level
- Allowed to ‘clear the cache’ and ‘read list of cached sectors’

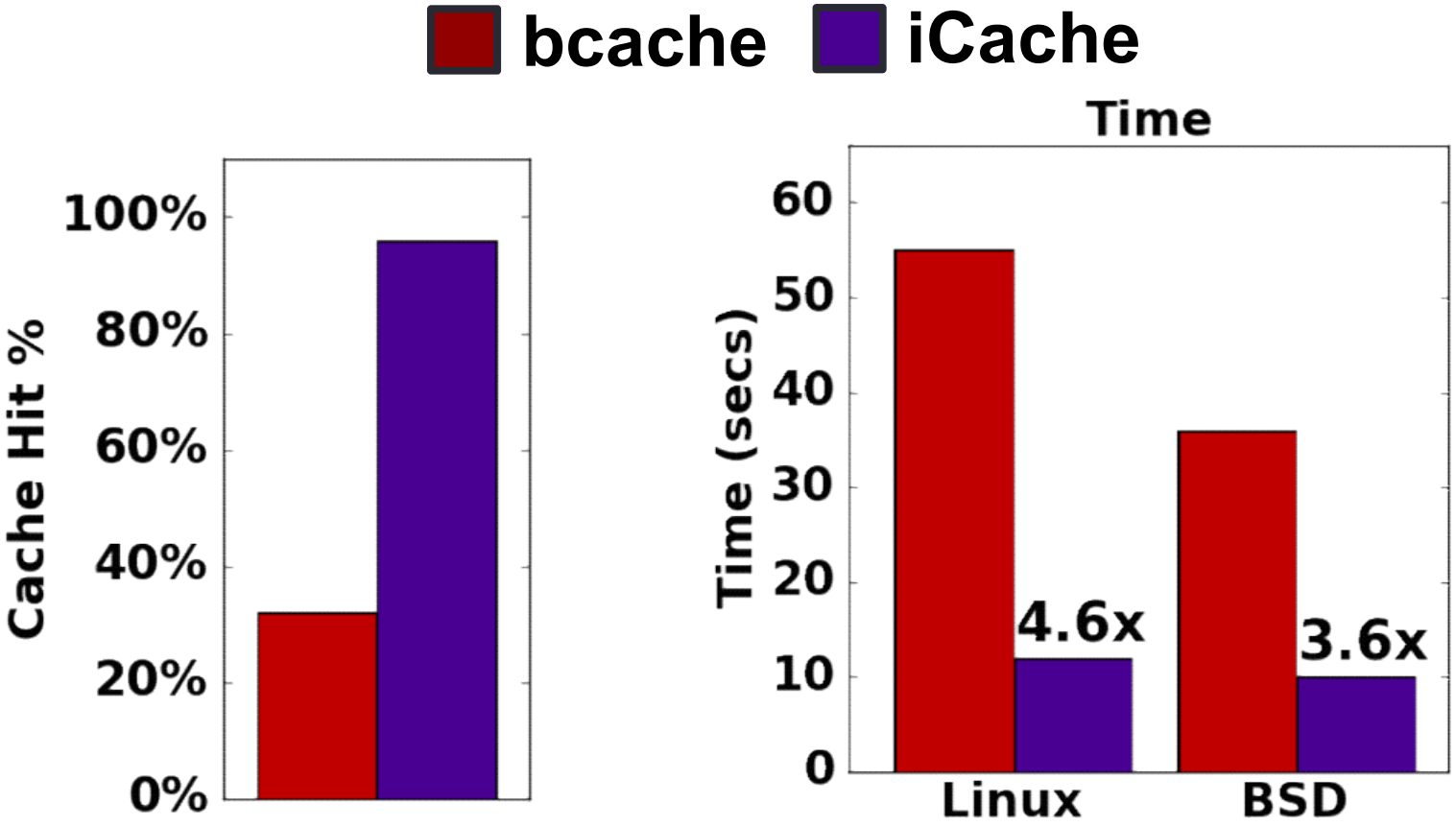
I/O classification policy:

File Size(MB)	> 10MB	< 10MB	< 5MB	< 2MB	< 1MB	Metadata
I/O Class	0 (default)	1	2	3	4	5

Find (File Name Search)

Is *iCache* effective for metadata only workloads? Yes

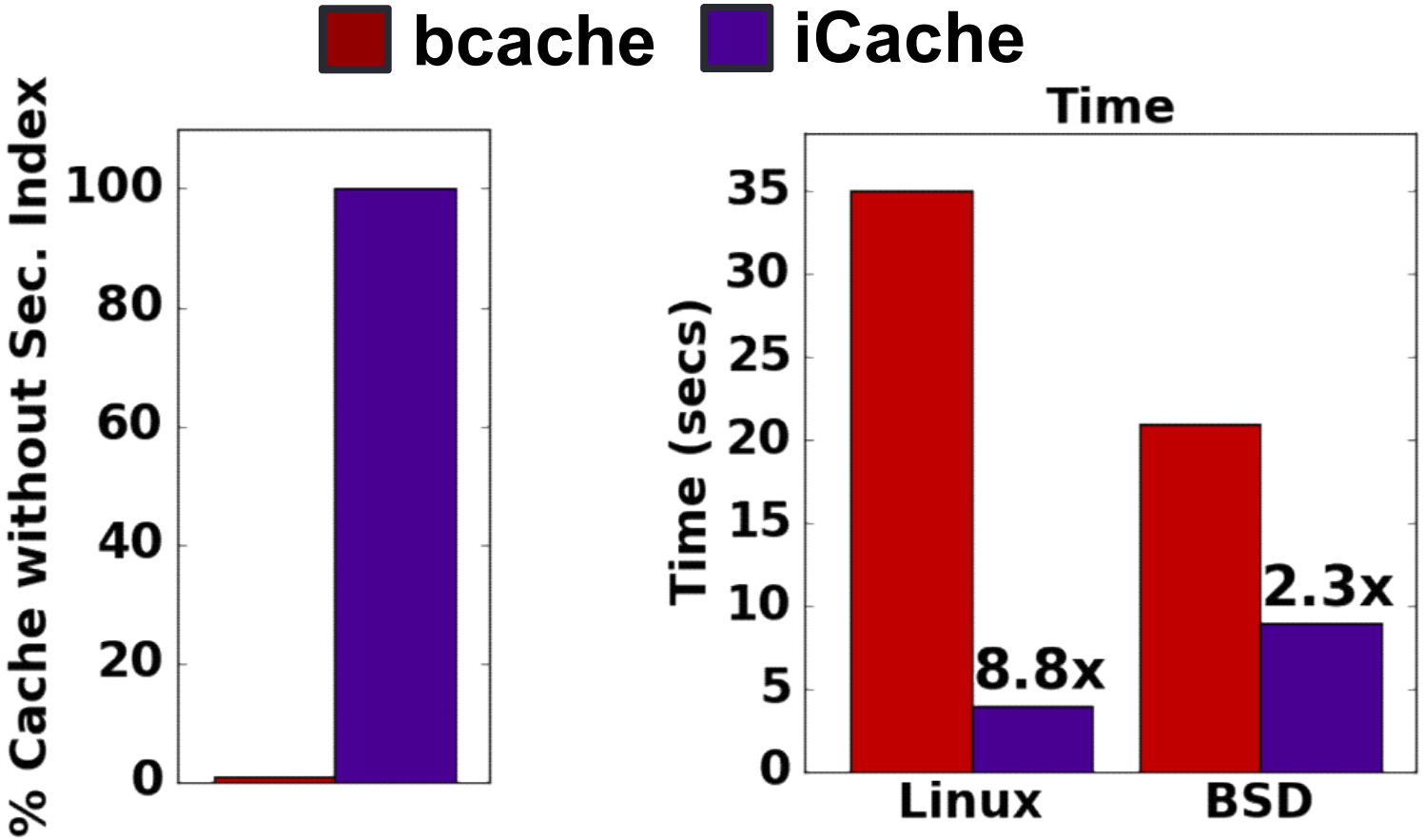
Search for non-existent file in 6GB of Linux kernel source code



TPC-H Join Query on MySQL

Is application supplied classification useful? Yes

- Higher priority to clustered index over secondary index
- Secondary index is huge in size and also sequential on disk



Case Study #3: iDedup

Dmdedup module with modifications (14.5% new code):

- Don’t dedup I/O requests that have unique content hints
- Cache disk-backed mappings for file-copy I/O pattern
- Statistics collection and reporting

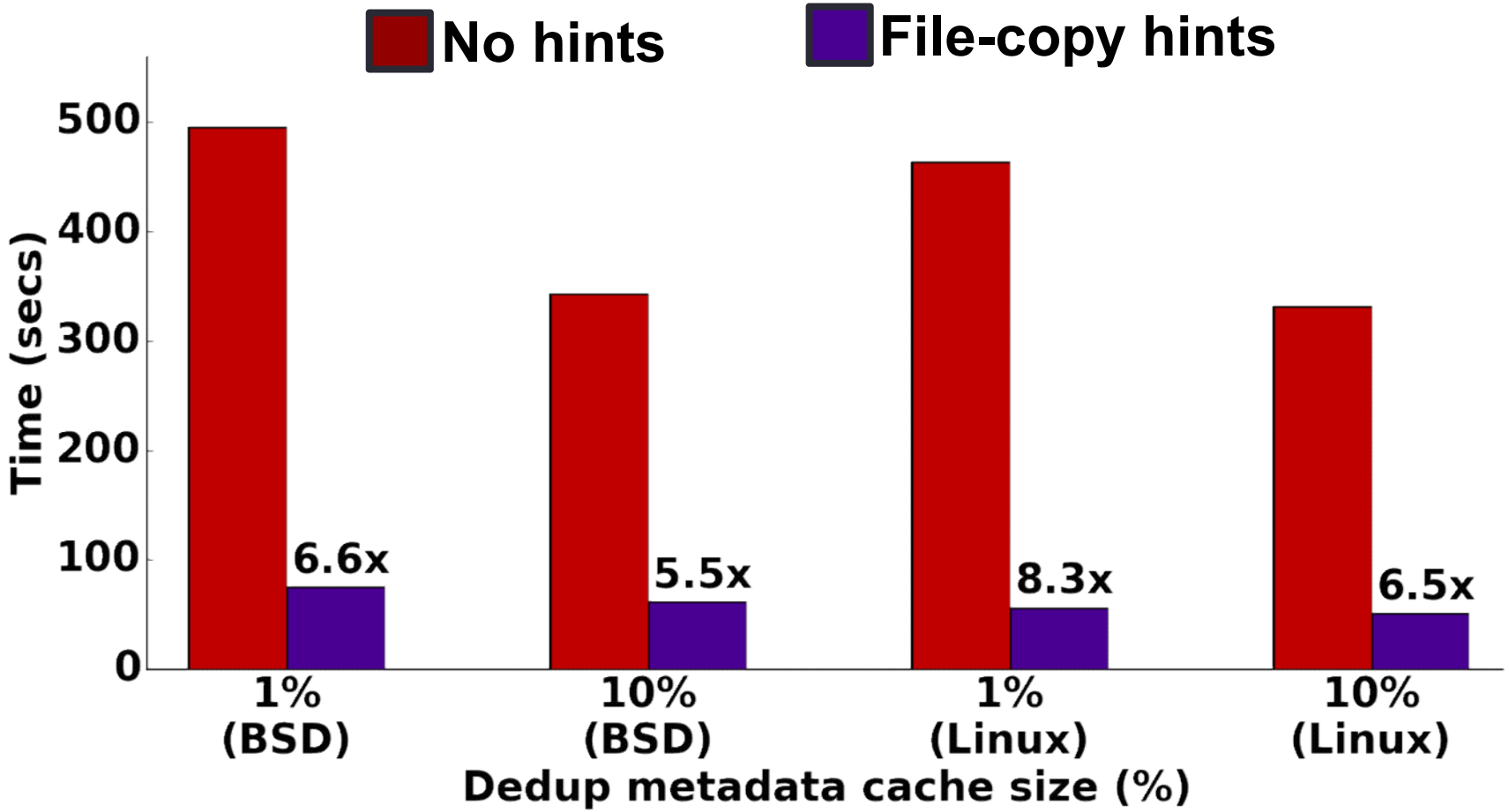
Dedup metadata cache size: 1% and 10% of total needed (Typical in real deduplication environments)

File-copy I/O pattern

Do file-copy hints result in runtime improvements? Yes

Avoids costly disk-backed hashtable lookups for writes

Copy 500 MB file containing random data

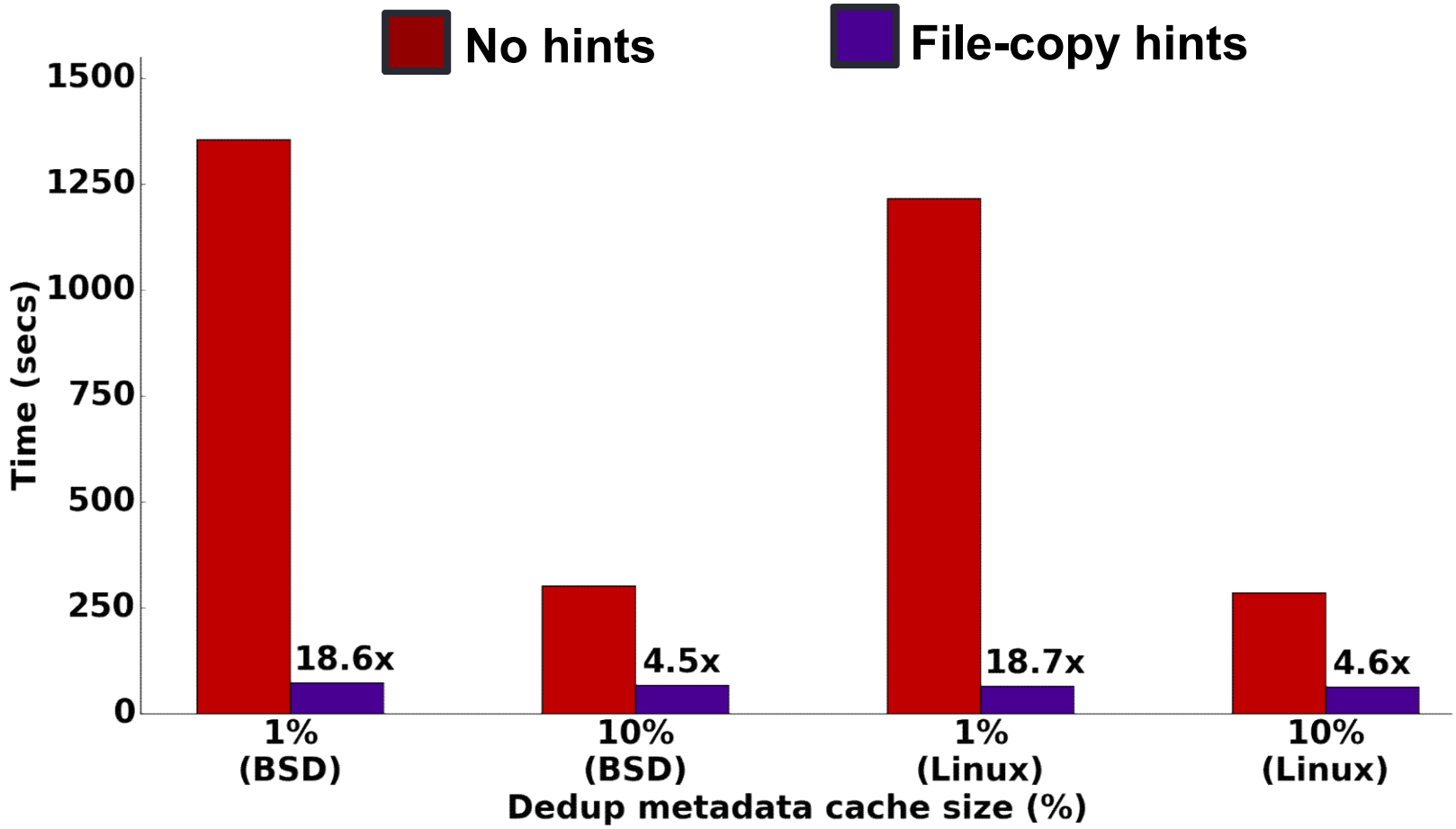


File Encryption

Are file encryption hints useful? Yes

Avoids deduplication of I/O with unique content

Encrypt 500 MB file containing random data



Conclusion

Semantic Information about guest I/O requests useful at VMM

Information can be obtained implicitly without modifications

Sky is a hypervisor extension that:

- gathers insights about guest I/O using system-call interception
- easy to deploy because it doesn't modify guest OS or FS
- has low overhead for real I/O workloads
- good classification accuracy

iCache gives higher priority to small files, metadata and application classified content [2.3x to 8.8x Imp.]

iDedup avoids deduplication for encrypted I/O and caches disk-backed mappings for file-copy I/O [4.5x to 18.7x Imp.]