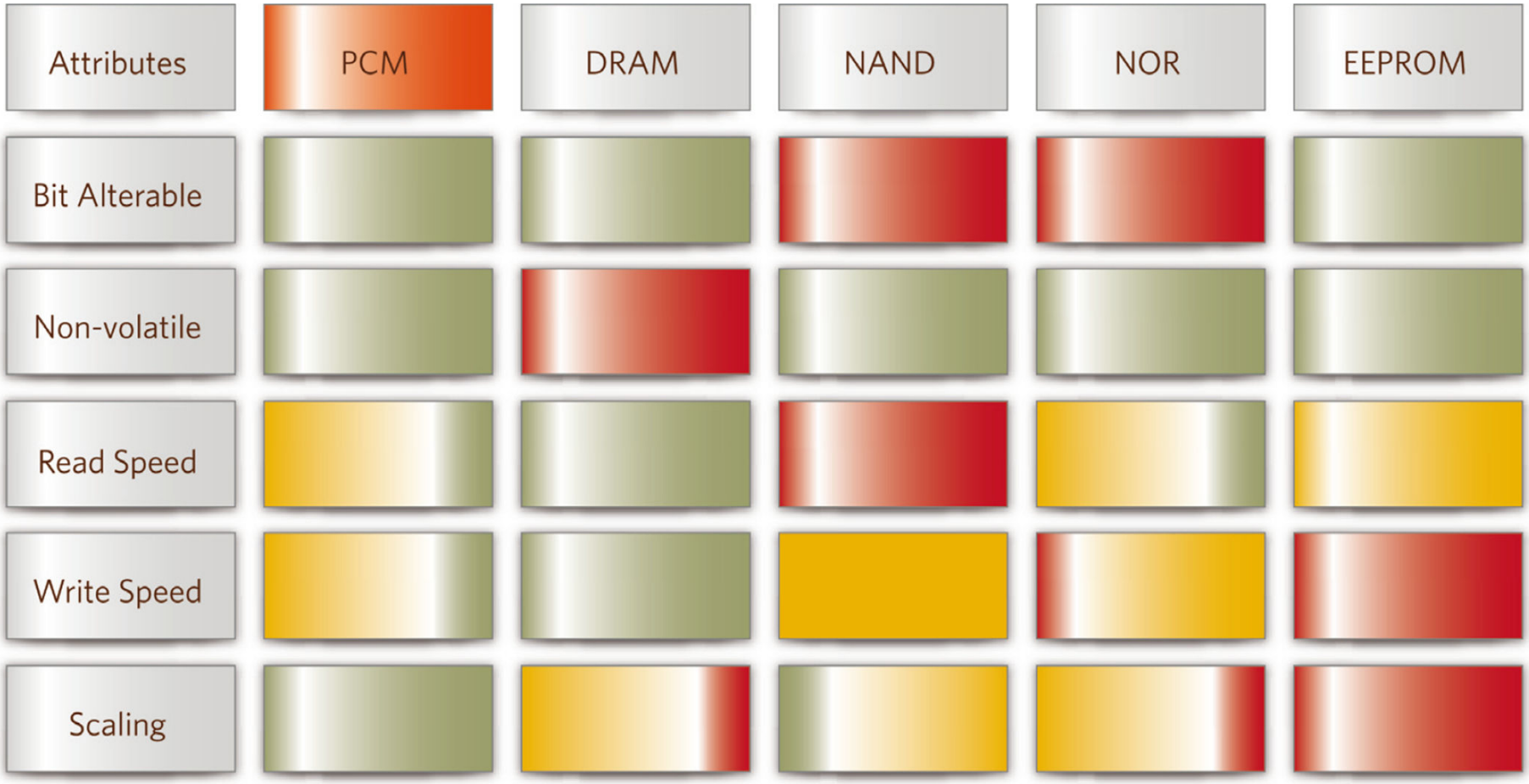
SCMFS: A File System for Storage Class Memory

Xiaojian Wu, Narasimha Reddy
Texas A&M University

What is SCM?

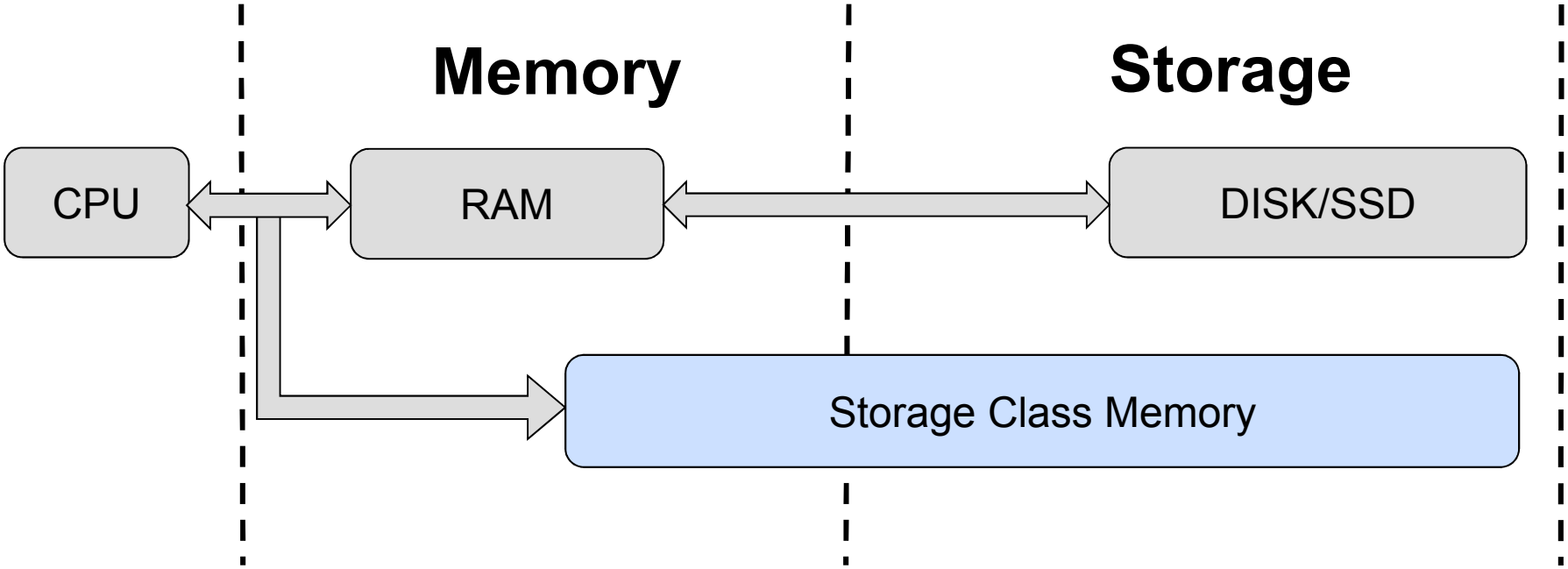
- Storage Class Memory
  - Byte-addressable, like DRAM
  - Non-volatile, persistent storage
- Example: Phase Change Memory

PCM Attributes



From Numonyx

Hardware hierarchy



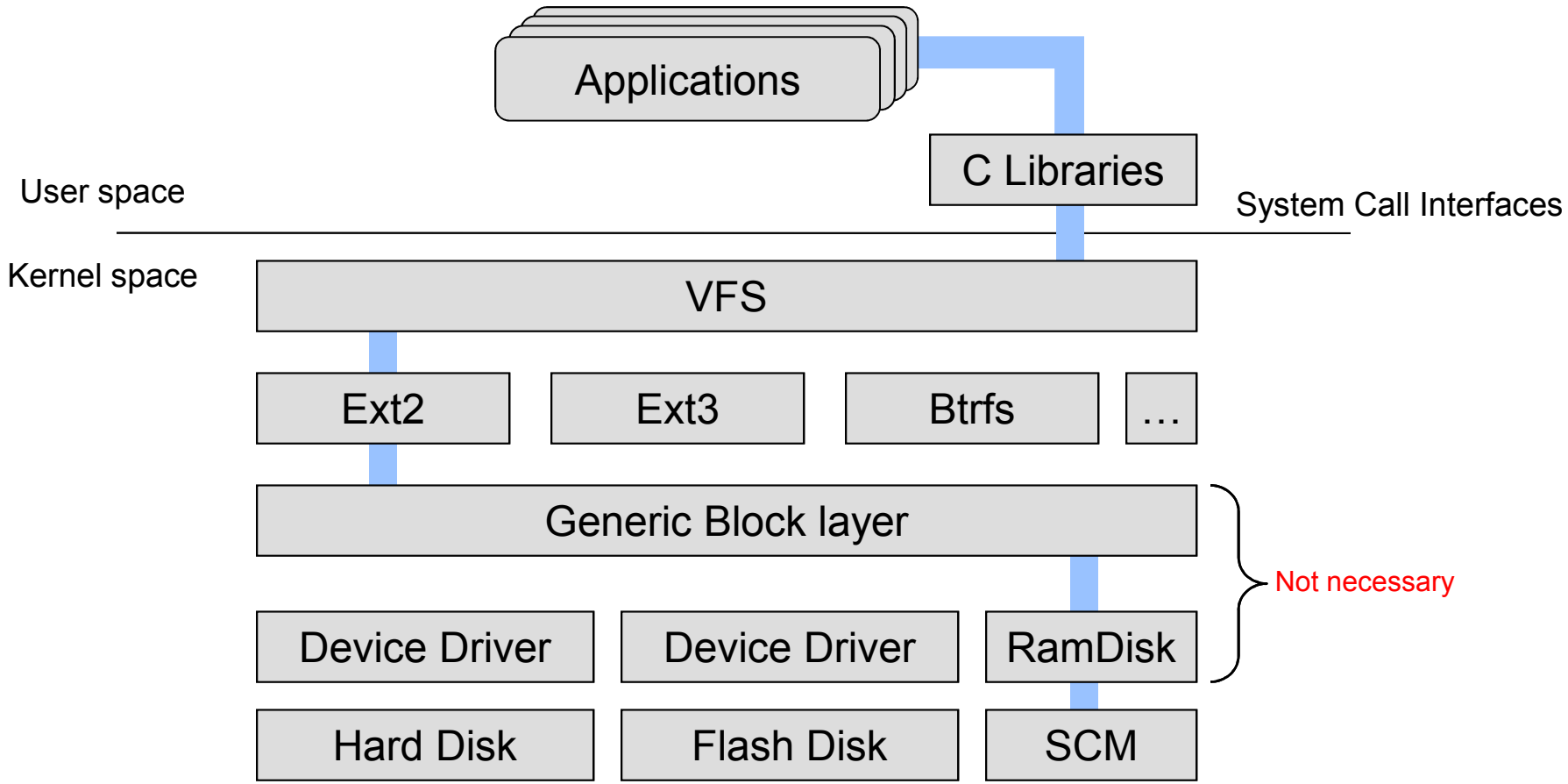
How to use SCM as storage?

- External interfaces change slower than internals.
  - Provide consistent APIs,
    - e.g., POSIX
  - Name space
    - Directory-based file systems

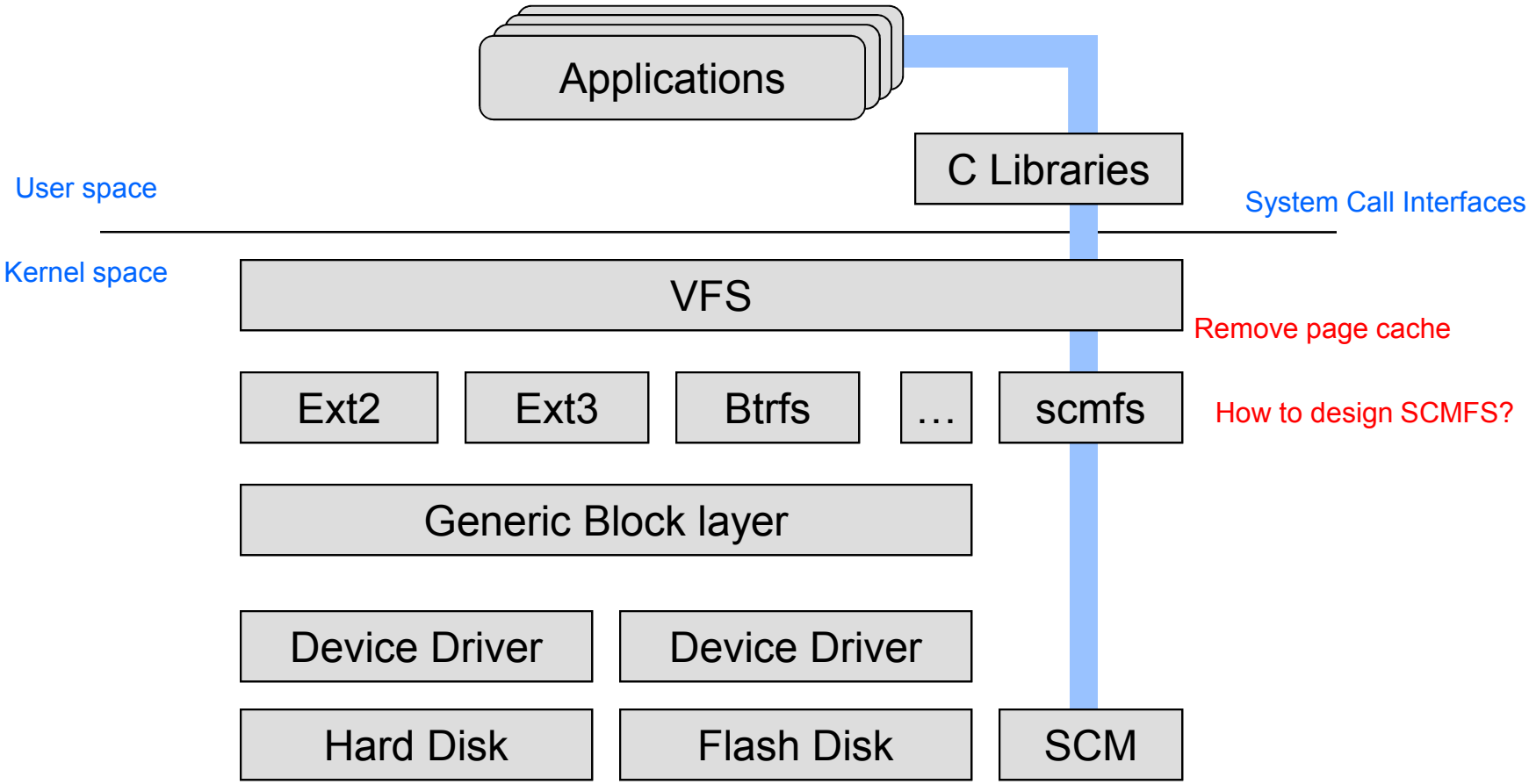
How to use SCM as storage?

- Device level
  - Use existing file system on RamDisk
- File system level
  - Design a new file system

Use existing FS on Ramdisk



New FS on SCM


Applications
C Libraries
User space
System Call Interfaces
Kernel space
VFS
Remove page cache
Ext2
Ext3
Btrfs
…
scmfs
How to design SCMFS?
Generic Block layer
Device Driver
Device Driver
Hard Disk
Flash Disk
SCM

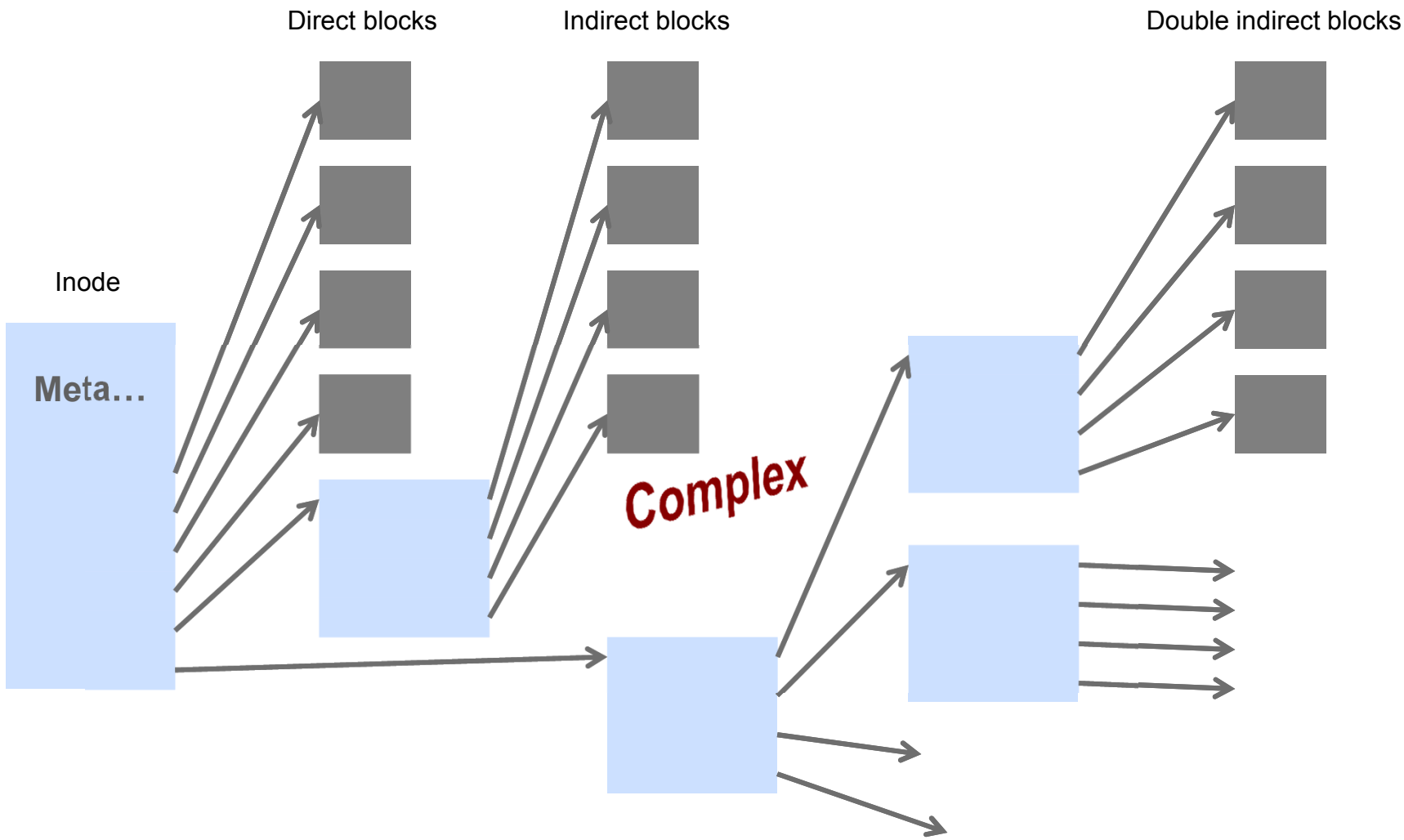
Keep It Small and Simple

Simplify the SCMFS

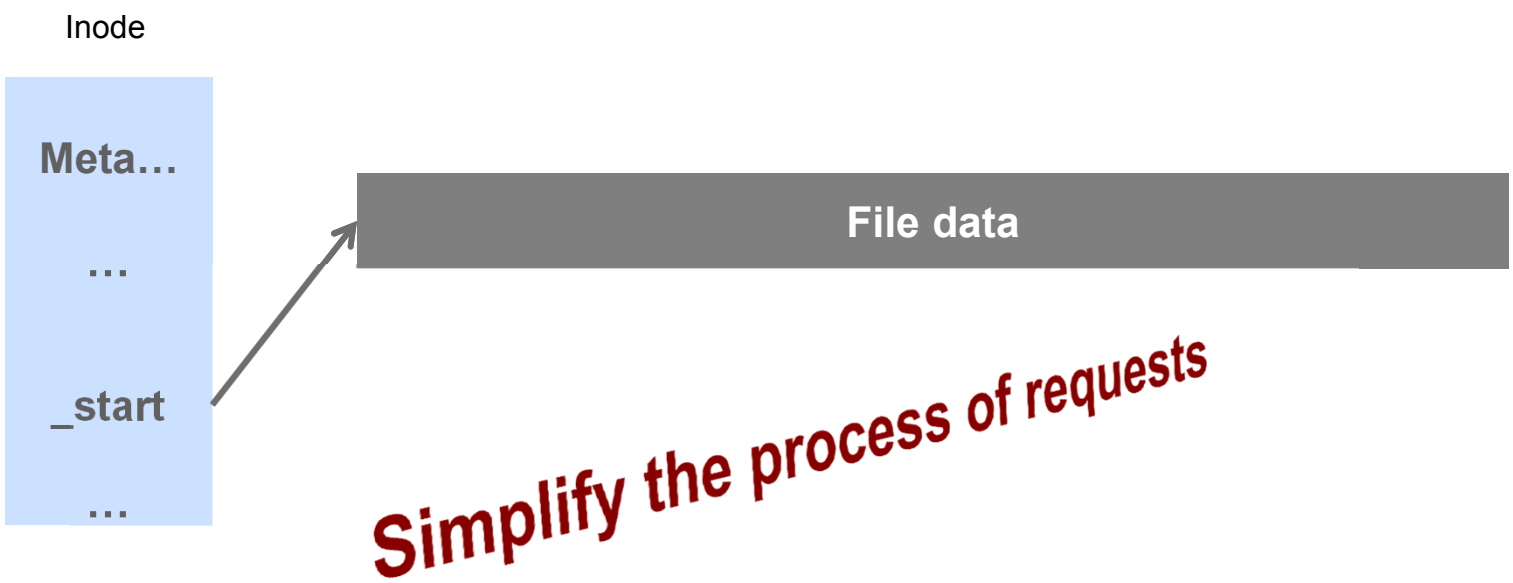
- **Block management in FS**
  - Allocate/De-allocate blocks
  - Manage the resources on storage device
- **Memory Management in OS**
  - Allocate/De-allocate memory pages
  - Manage the memory resources

Similar! Take advantage of MMU.

Indirect blocks in regular FSs



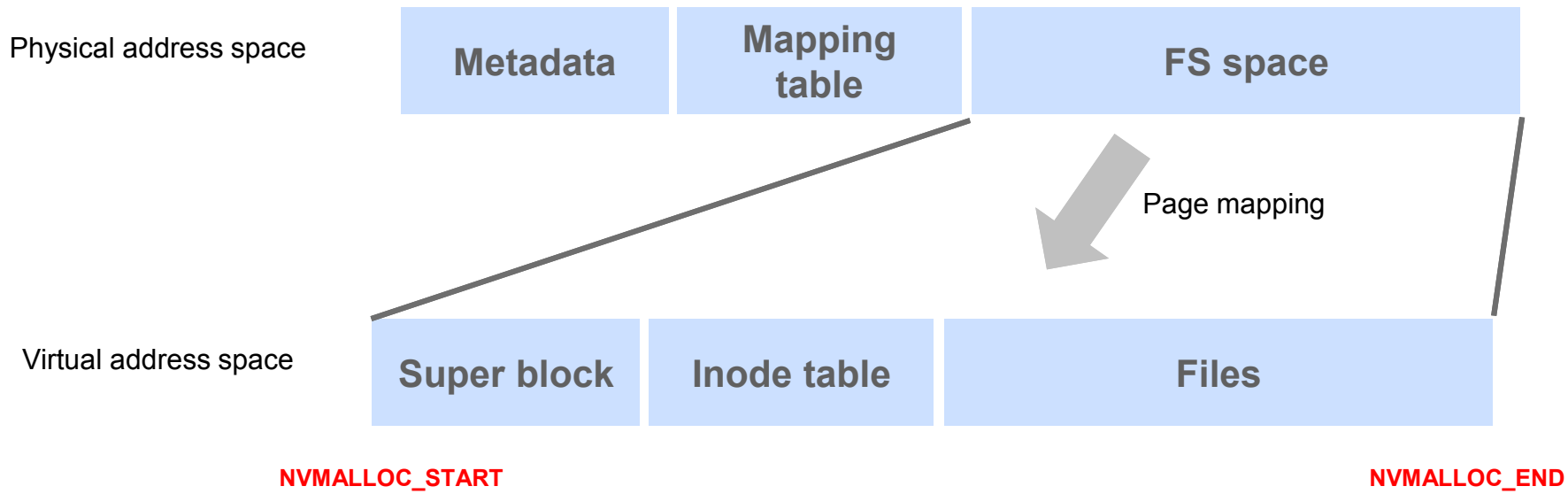
Keep files always contiguous



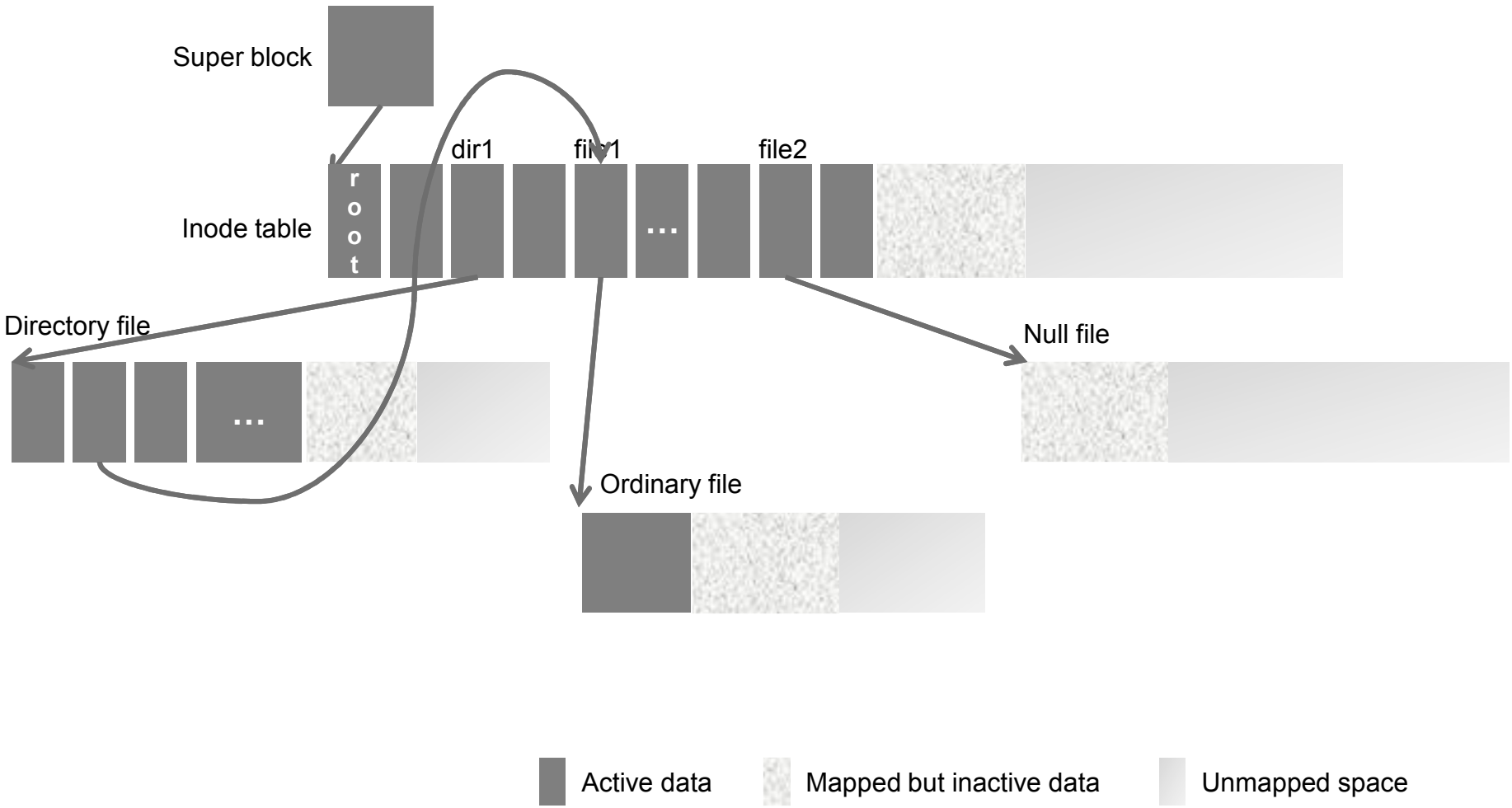
Wrap up

- Re-utilize Memory Management (MM) module in O/S to do block management
- Implement the file system in Virtual Address Space
- Keep all the files contiguous in Virtual Address Space

Memory space layout



SCMFS File system layout



Modify OS kernel to support SCMFS

- OS should be able to distinguish SCM from volatile DRAM
  - Add a new address range type "AddressRangeStorage" to E820 table
- Add a new memory zone "ZONE STORAGE"
  - Put memory range with "AddressRangeStorage" into this zone
- Add new kernel APIs
  - nvmalloc()/nvfree()/nvmalloc_expand()/nvmalloc_shrink()…
  - Always operate in the zone "ZONE STORAGE" (physical address space)
  - Always operate between **NVMALLOC_START** and **NVMALLOC_END** (total 2^{47})

More features

- Pre-allocation
  - Allocate more space than needed
  - Null files
  - Garbage collection
- File system consistency
  - Write ordering problem
    - Cache, CPU instruction re-ordering
  - Use clflush_cache_range to provide metadata consistency
    - Combination of MFENCE and CLFLUSH.
  - Flush the CPU cache periodically to provide data consistency.

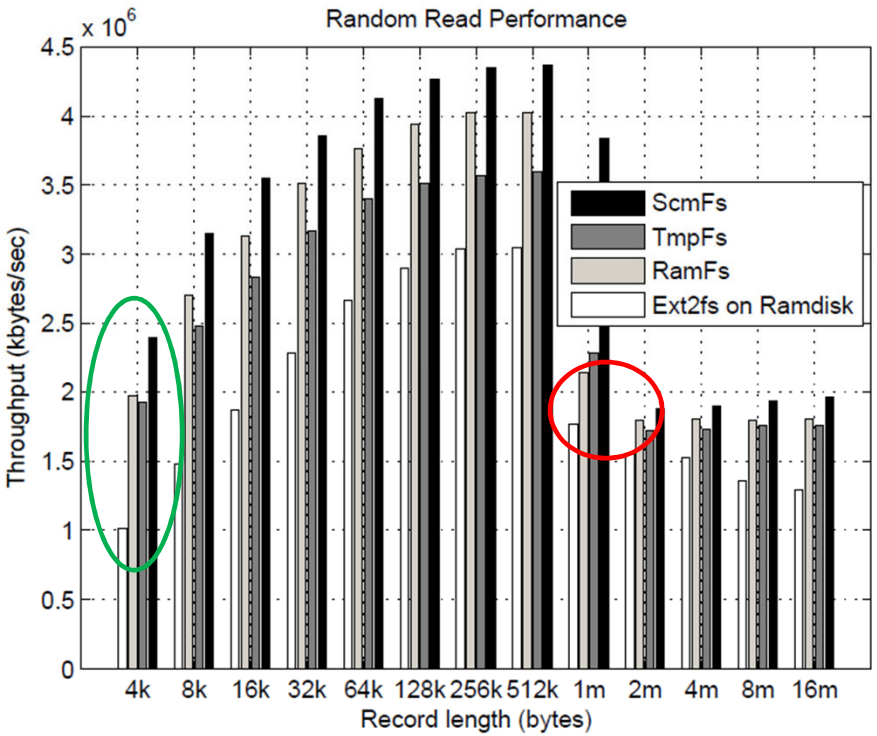
Evaluation

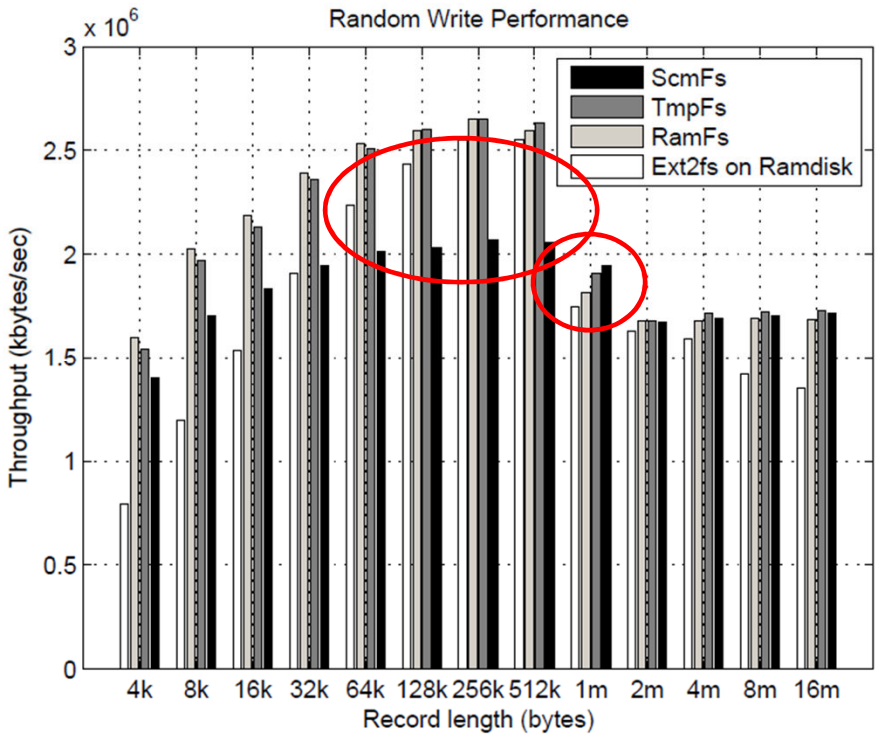
- Environment
  - 2.33GHz Intel Core2 Quad Processor Q8200
  - 8GB RAM, 4GB is used as SCM.
  - Linux 2.6.33
- Benchmarks
  - IoZone – Workload on file data
  - Postmark – Workload on metadata
- Use Performance Counter to analyze the results

Simplicity

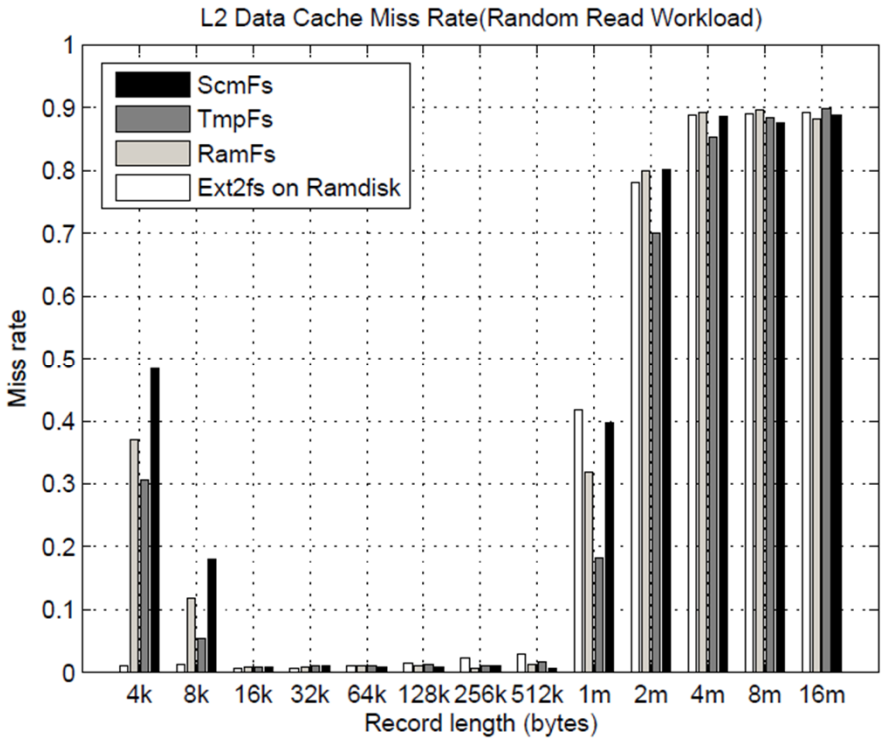
- Modification to OS consists of 300 SLOC.
- SCMFS consists of 2700+ SLOC (1/10th of Ext2FS)

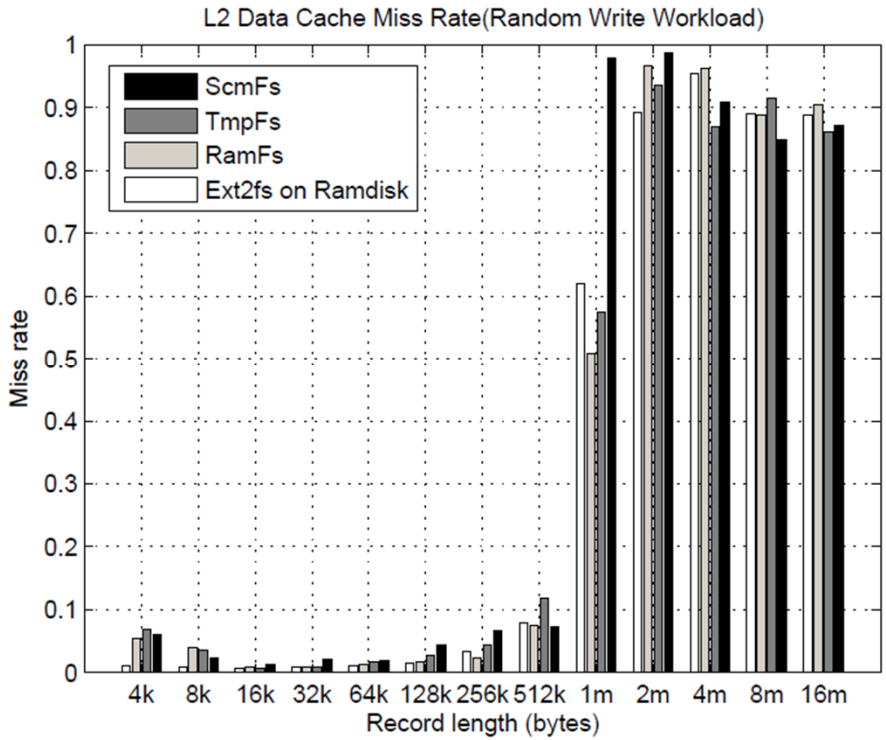
Performance (IoZone Random)



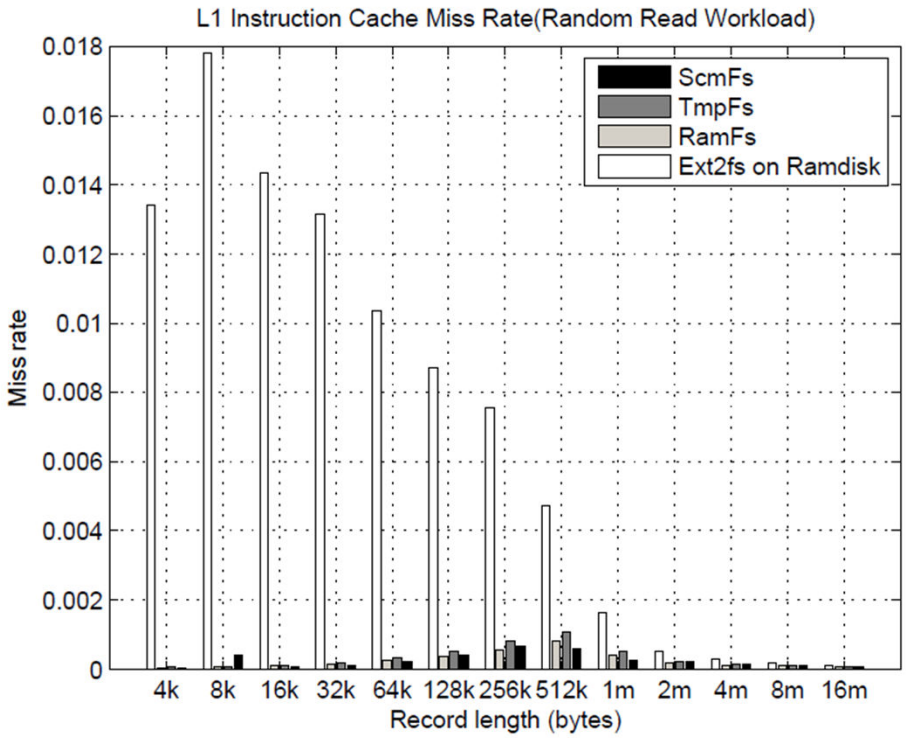


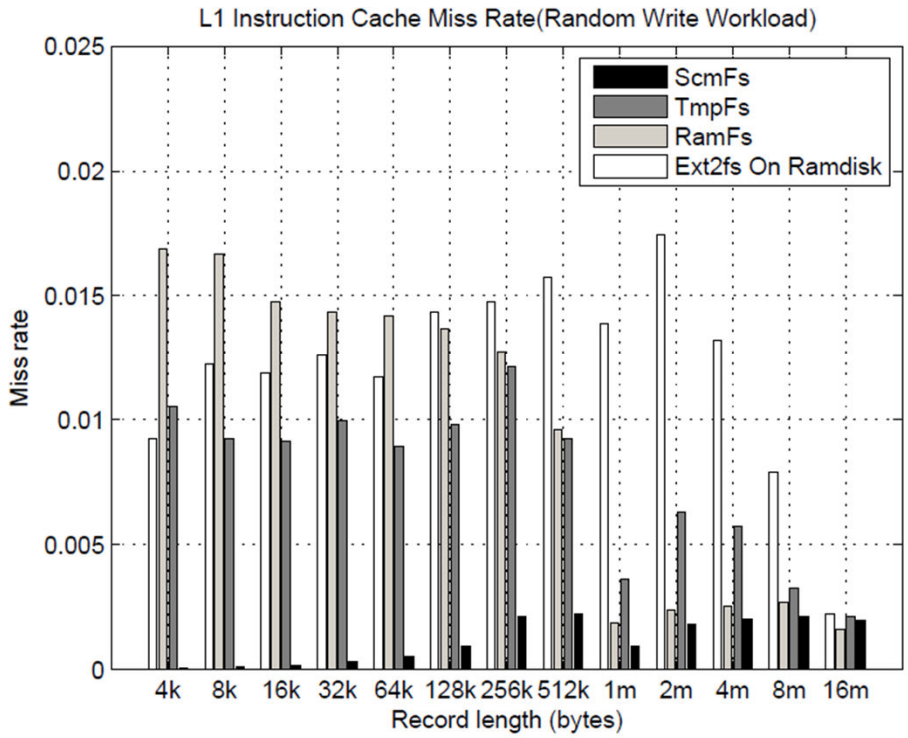
L2 Data Cache Miss Rate (IoZone Random)



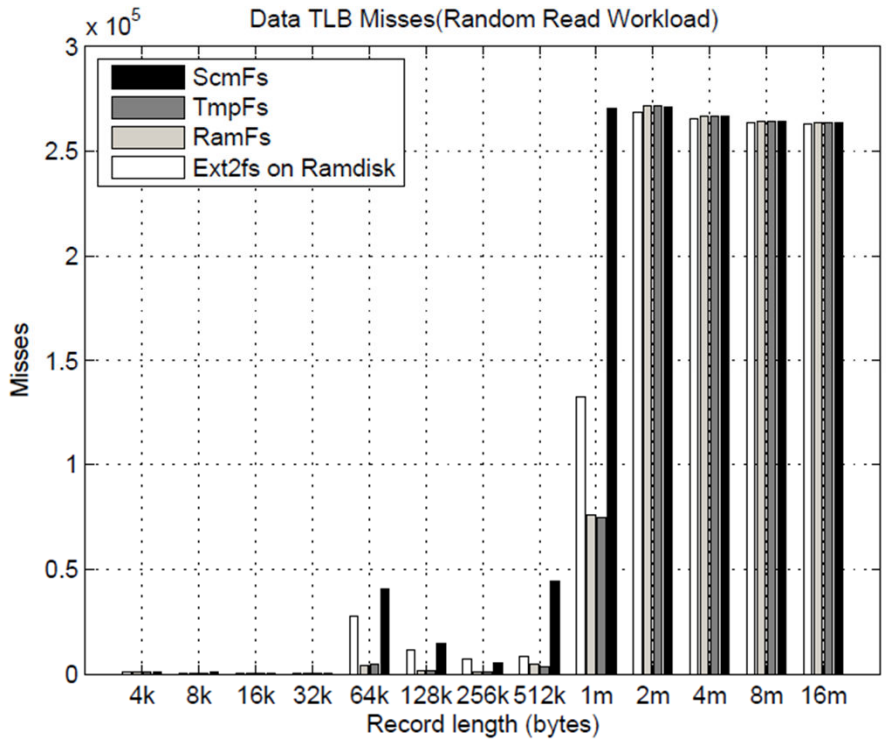


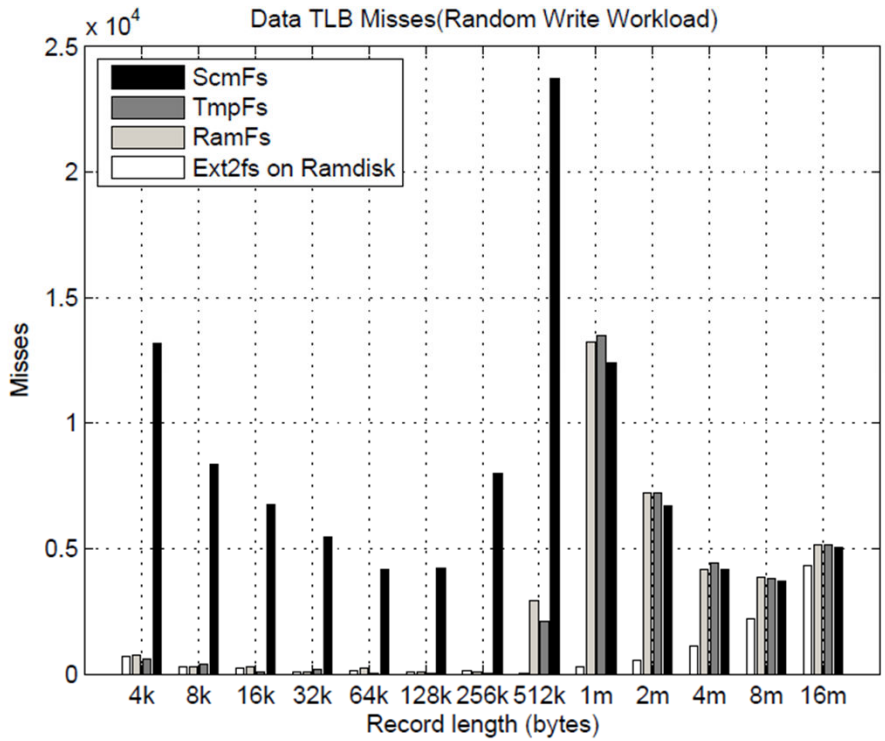
Instruction Cache Miss Rate (IoZone Random)





Data TLB Misses (IoZone Random)





SCMFS suffers from TLB misses.

Why higher TLB misses in SCMFS?

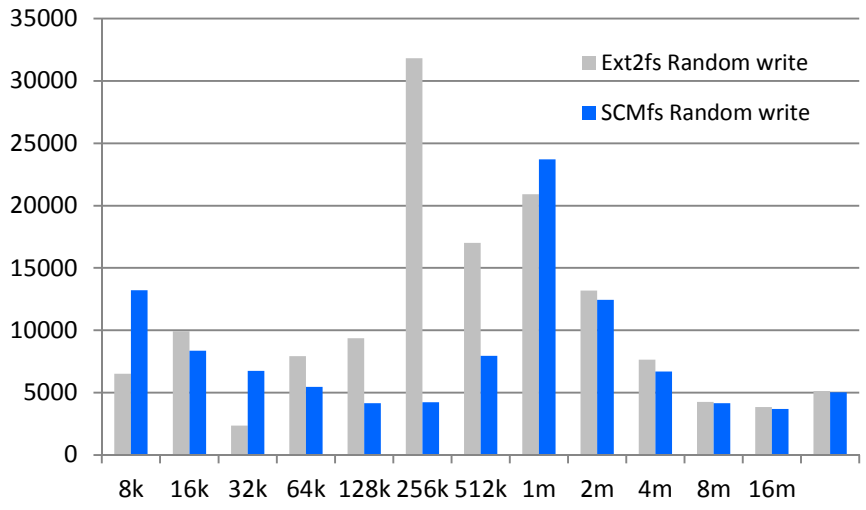
Scmfs works here, use small page size (4K)

RamDisk works here, use big page size(2M)

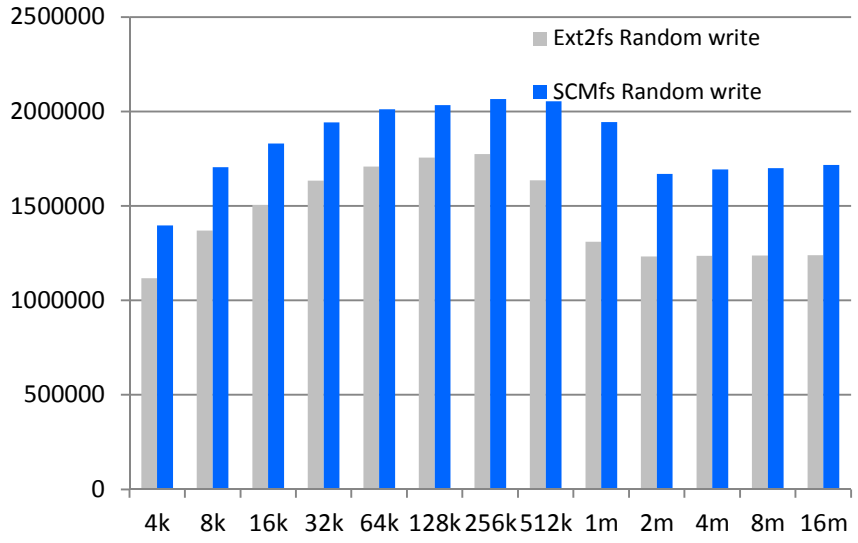
Memory Map (x86_64)

(=47 bits) user space	0000000000000000 - 00007fffffffffff
hole caused by [48:63] sign extension	
(=47 bits)nvmalloc space	**ffff000000000000 - ffff7fffffffffff**
(=40 bits) guard hole	ffff800000000000 - ffff80ffffffffff
(=64 TB) direct mapping of all phys.	ffff880000000000 - ffffc7ffffffffff
(=40 bits) hole	ffffc80000000000 - ffffc8ffffffffff
(=45 bits) vmalloc/ioremap space	**ffffc90000000000** - ffffe8ffffffffff
(=40 bits) hole	ffffe90000000000 - ffffe9ffffffffff
(=40 bits) virtual memory map (1TB)	ffffea0000000000 - ffffeaffffffffff
(=512 MB) kernel text mapping, from phys 0	**ffffffff80000000** - ffffffffa0000000
(=1536 MB) module mapping space	ffffffffa0000000 - ffffffffff00000

Why higher TLB misses in SCMFS?



Data TLB Misses



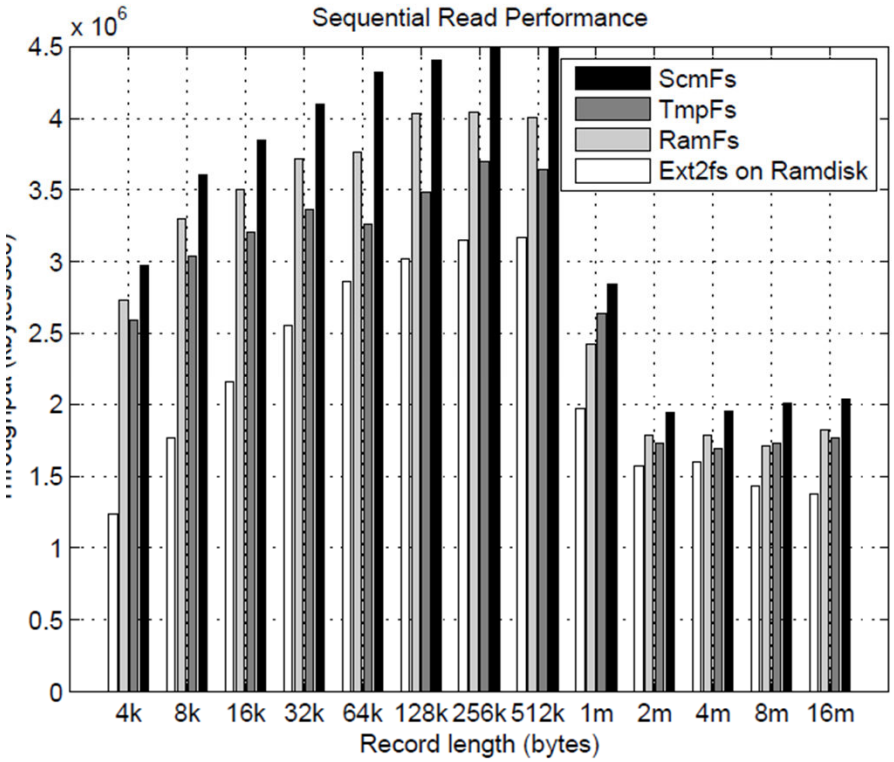
Throughput(kbytes/s)

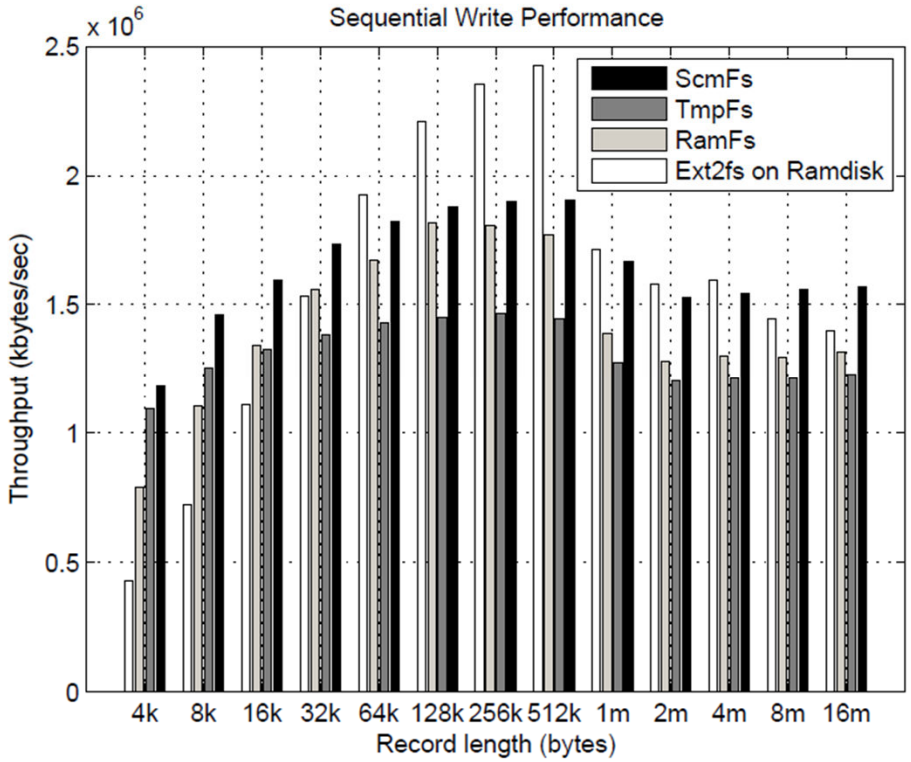
- **Disable page size extension to use 4k page size everywhere**

How to reduce Data TLB Misses?

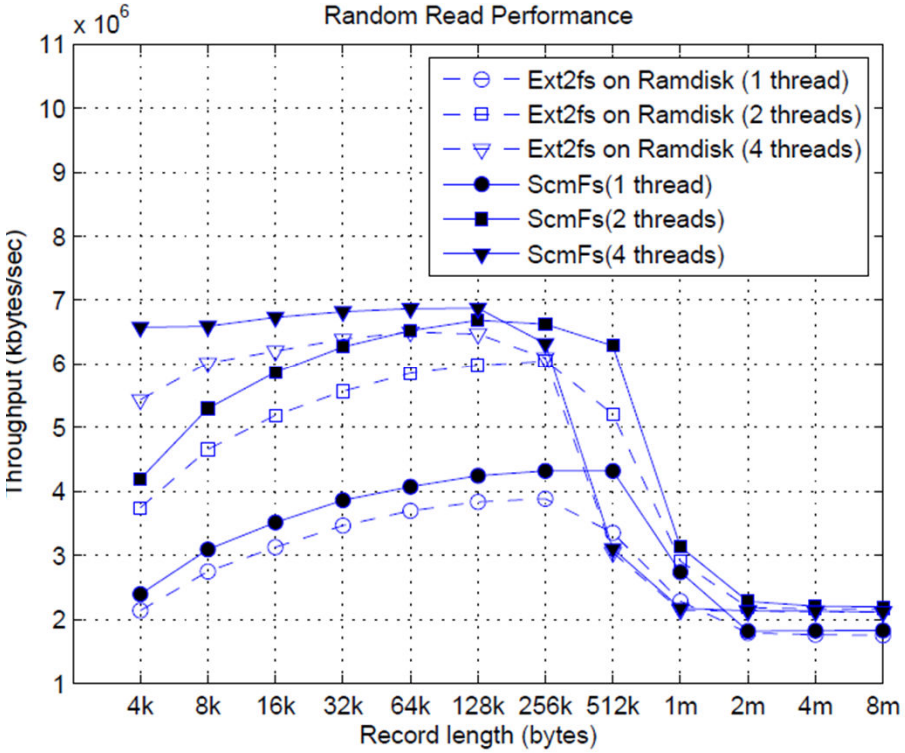
- Larger page size
- Use linear mapping address for small files (<4kbytes)
- Pre-allocate huge page and manage space inside huge page

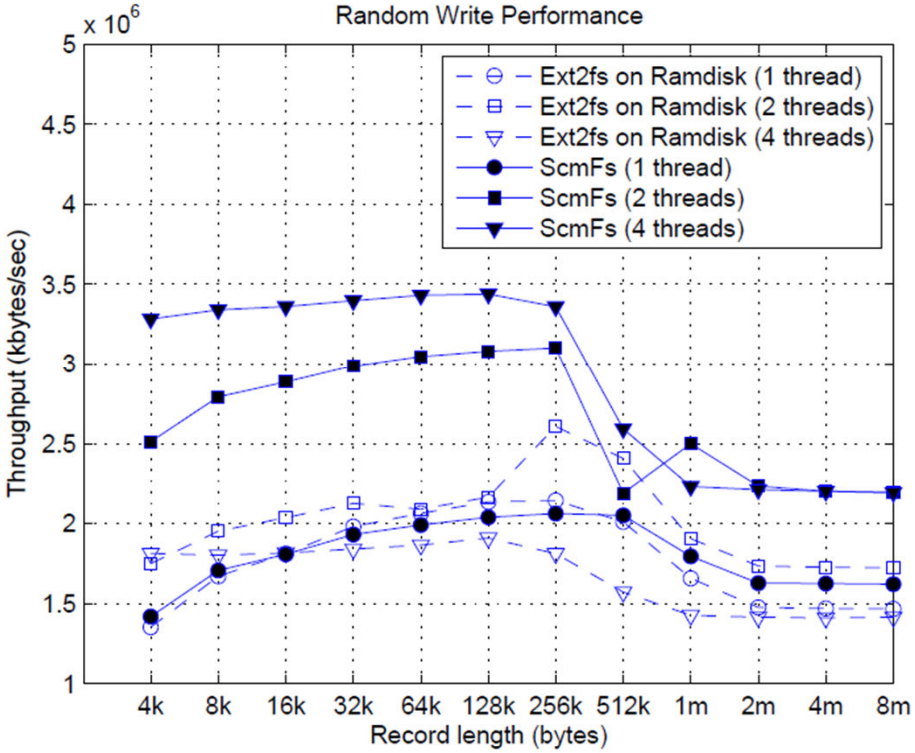
Performance (IoZone Sequential)





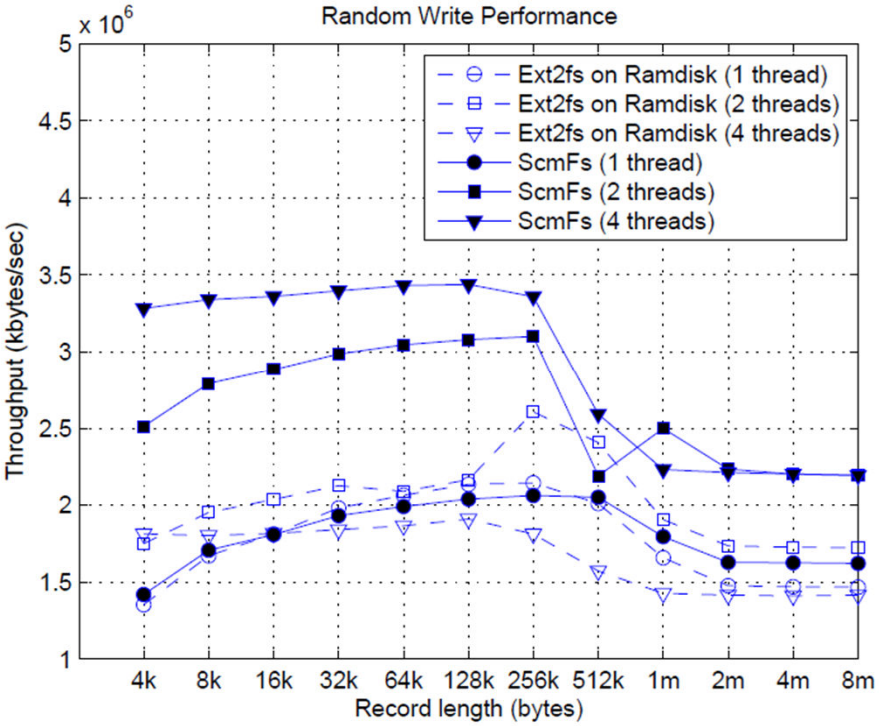
Multi-thread Performance (IoZone Random)



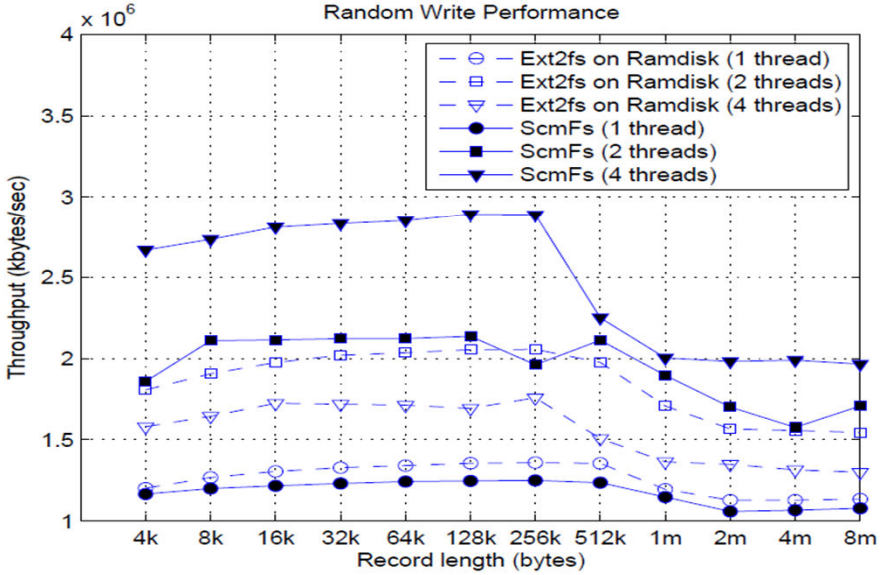


Does XIP mmap() perform best?

Multi-thread Performance (IoZone Random)



read()/write()



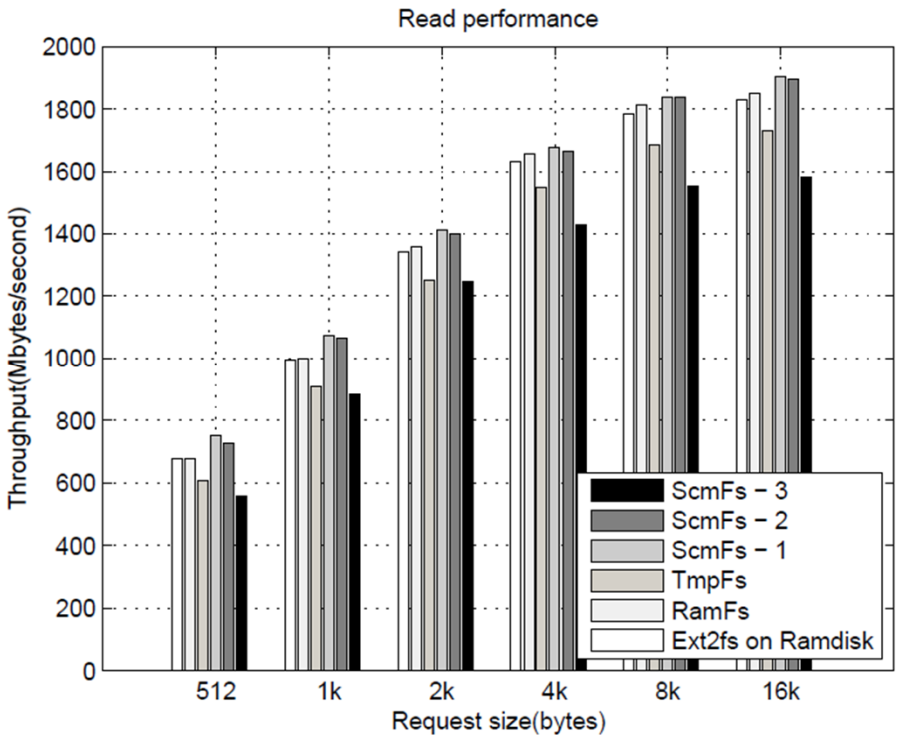
mmap()/bcopy()

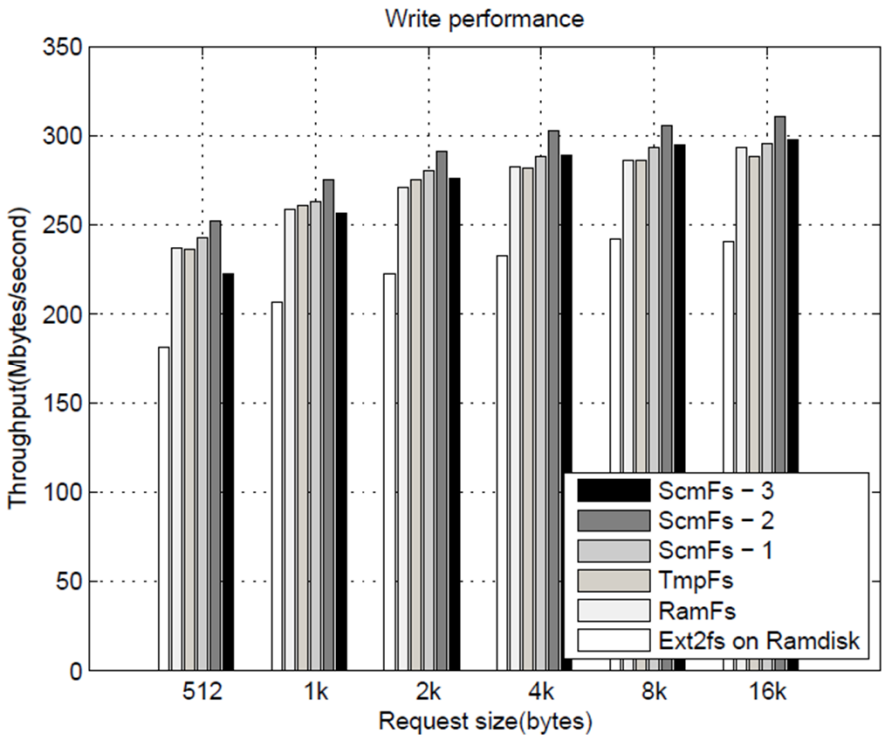
TLB misses

read()/write() - 200(Ext2fs) or 4,000(SCMFS)

mmap() - 2,000,000

Performance (postmark Random)





SCMFS-1, original SCMFS.
SCMFS-2, SCMFS-1 with space pre-allocation.
SCMFS-3, SCMFS-2 with file system consistency.

Future work

- Reduce TLB misses.
- Simplify metadata operations.
- Defragmentation of virtual address space.
- Protection from malicious wearing out.

Related work

- A lot of papers on how to build large capacity main memory by using PCM
- BPFS
  - A file system designed for non-volatile byte-addressable memory.
  - Uses shadow paging techniques to provide fast and consistent updates.
  - Requires architectural enhancements.

Conclusion

- SCMFS is designed for Storage Class Memory.
- SCMFS takes advantage of MMU.
- Design of File System should adapt to the change of hardware hierarchy.
- Performance depends on much more factors than ever.

Thanks