TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science, Institute for System Architecture, Operating Systems Group

FS^2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption

Hai Huang, Wanda Hung, Kang G. Shin

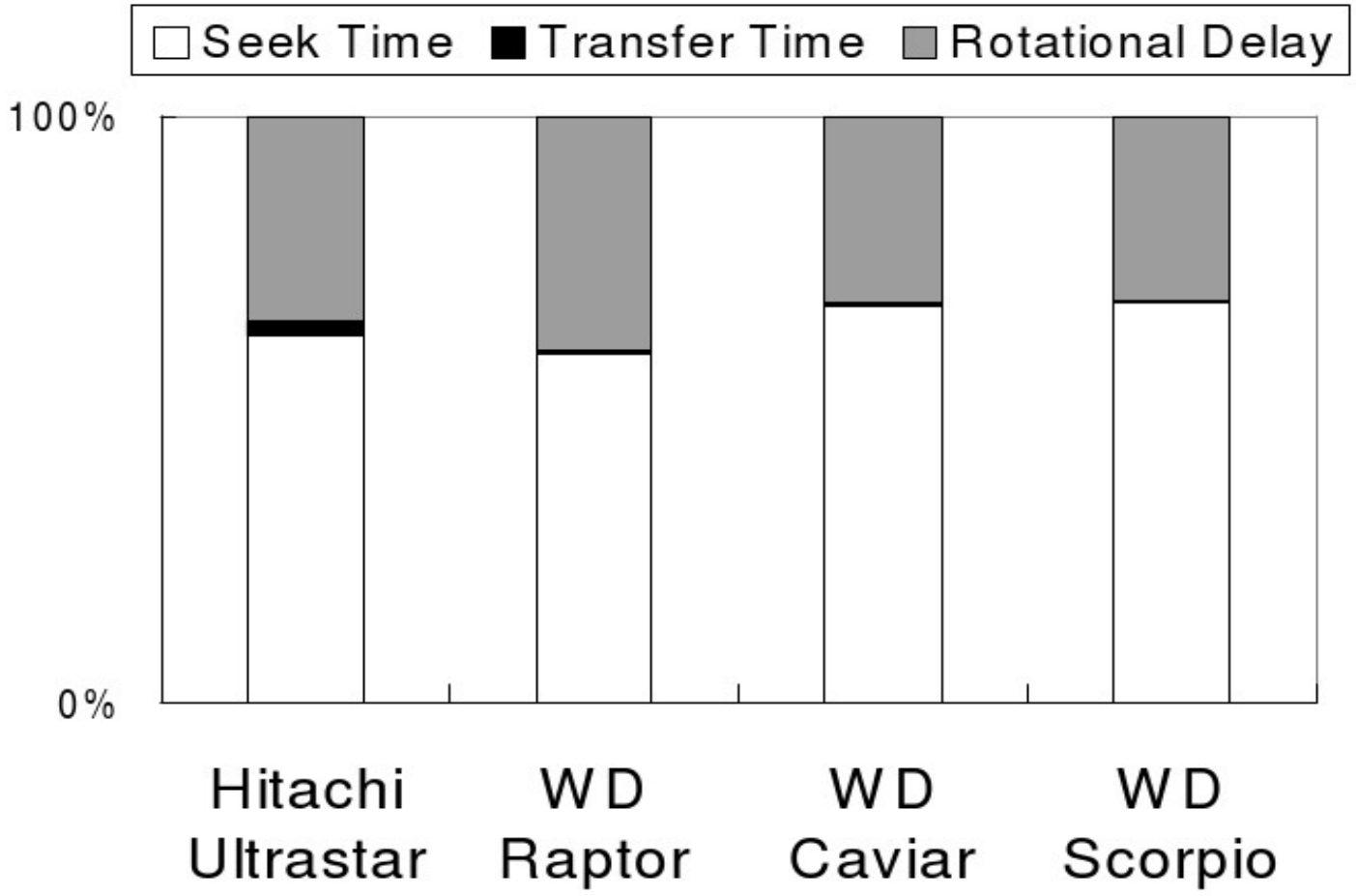
Presented by Carsten Weinhold

Paper Reading Group, 2008-09-18

Motivation

- Many aspects of magnetic disks improved, except for access times
- Seek times / rotational delay can reduce throughput considerably
- File systems try to minimize seeking (e.g., cylinder groups in FFS, Ext2, ...)
- Works for many workloads, but:
  - Static decision on data placement
  - No knowledge about future access patterns
  - Access locality sometimes impossible

Breakdown of Disk Access Time



CVS Update: Seeking Activities

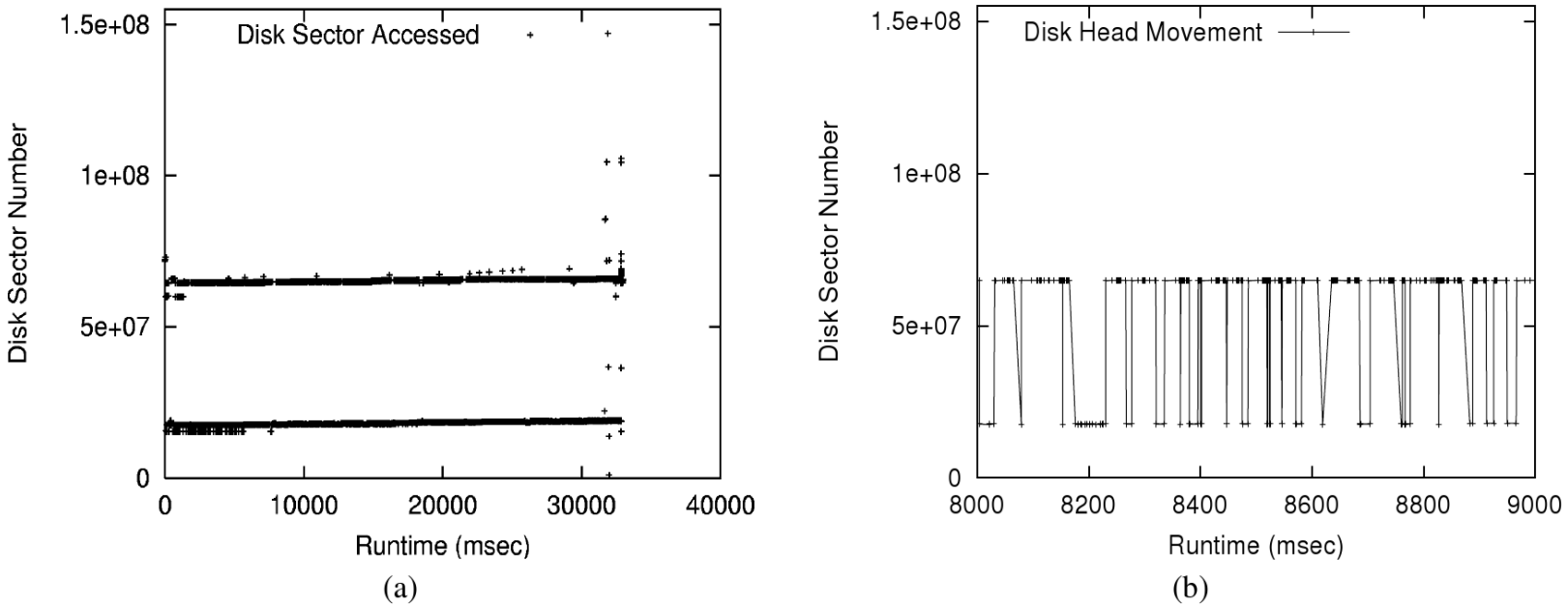
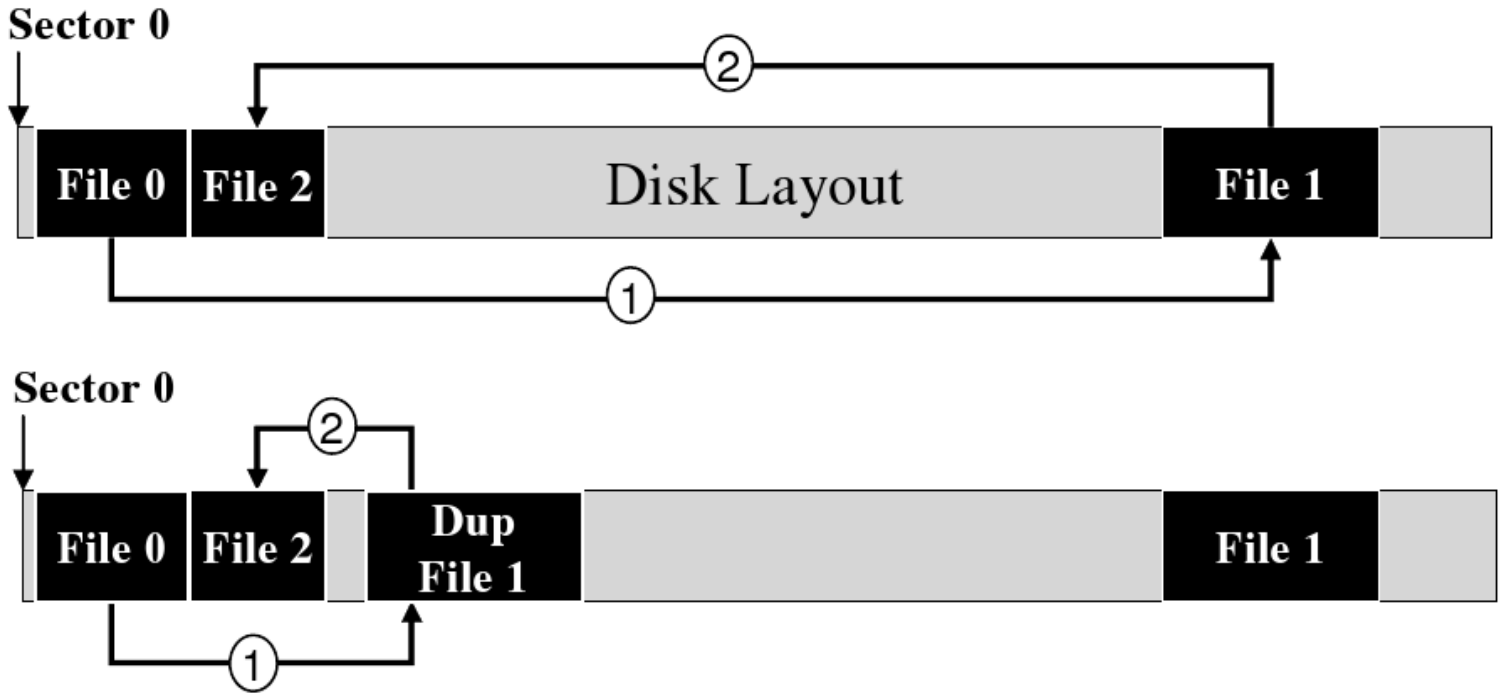


Figure 1: Part (a) shows disk sectors that were accessed when executing a *cvs -q update* command within a CVS local directory containing the Linux 2.6.7 source code. Part (b) shows the disk head movement within a 1-second window of the disk trace shown in part (a).

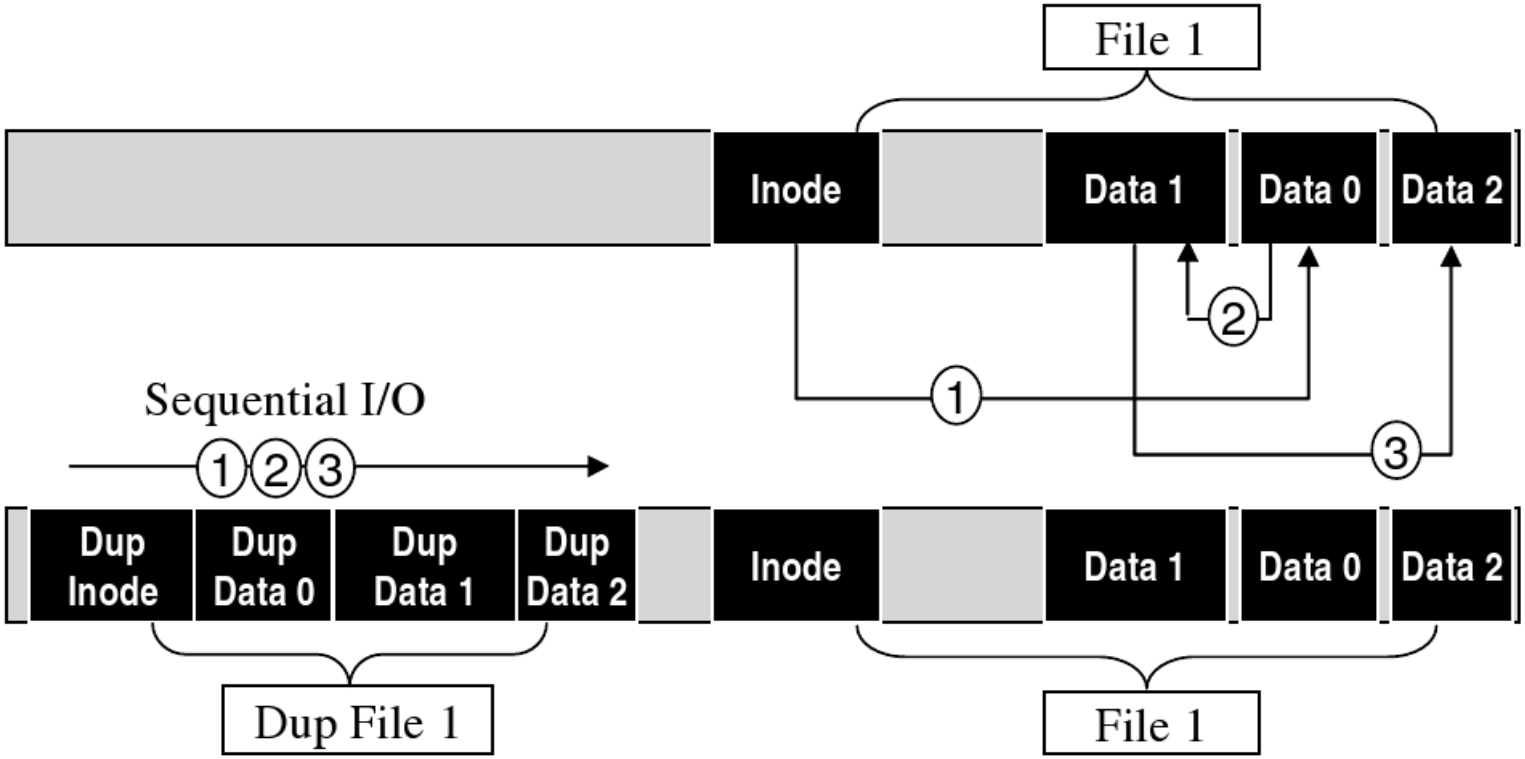
Idea

- Manage disk layout dynamically like other resources
- "Reallocate" data and meta data based on observed access patterns
- Keep copies of disk blocks in free space, there's plenty of it!
- Seek to closest copy relative to disk heads
- Benchmark workload "**`cvs update`**":
  - Ext2: ***33*** *seconds*
  - FS^2: ***22*** *seconds*
  - Less seeking, less energy

Approach


Sector 0
2
File 0
File 2
Disk Layout
File 1
1
Sector 0
2
File 0
File 2
Dup
File 1
File 1
1

Approach (II)


File 1
Inode
Data 1
Data 0
Data 2
2
1
3
Sequential I/O
1 2 3
Dup Inode
Dup Data 0
Dup Data 1
Dup Data 2
Inode
Data 1
Data 0
Data 2
Dup File 1
File 1

Managing Replicas

- Hash table for info about replicas:
  - Each entry stores:
    - Original block number
    - Replica block number
    - Last access time
    - Reference count
  - Kept in kernel memory
  - Flushed to disk periodically and on unmount
- Replicas are freed when:
  - Blocks are written to (no expensive updates)
  - Low on disk space

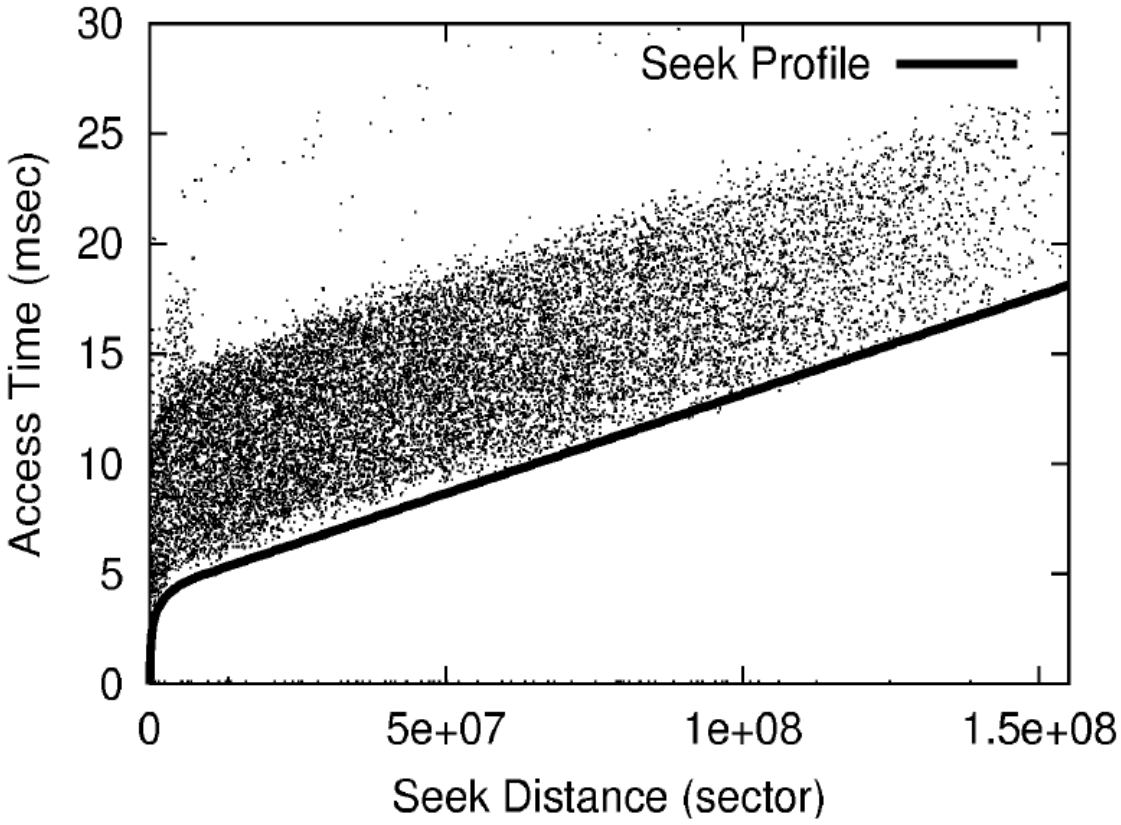
Implementation

- Modified *anticipatory I/O scheduler:*
  - Read original if there is no replica
  - Read replica if:
    - Blocks contiguous on disk
    - Closer to disk head than original
- Block device driver decides which blocks to replicate:
  - Temporally related blocks are good candidates
  - Identifies *hot areas* with much disk activity
  - Outside blocks replicated into hot area

Implementation (II)

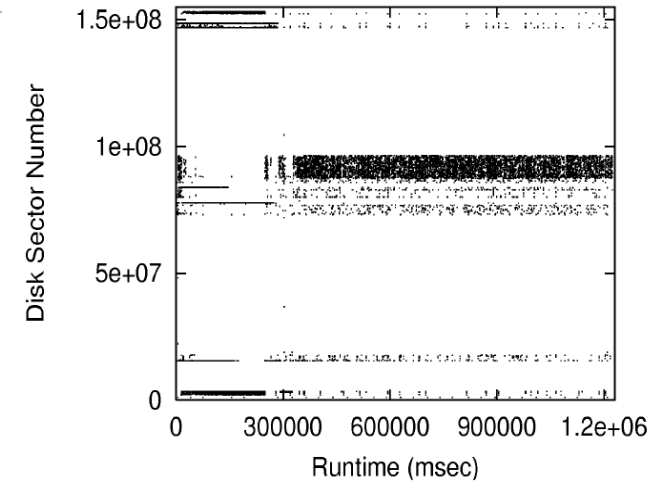
- Modified Ext2 file system:
  - Assists block device driver in finding free space for replicas
  - Notifies block device driver about deallocated / truncated blocks
  - Monitors high, low, and critical watermark
  - Maintains special file with inode #9 containing persistent hash table
- User-level tools
  - mkfs2, chkfs2
  - Explicitly control replica management

Decomposing Access Times

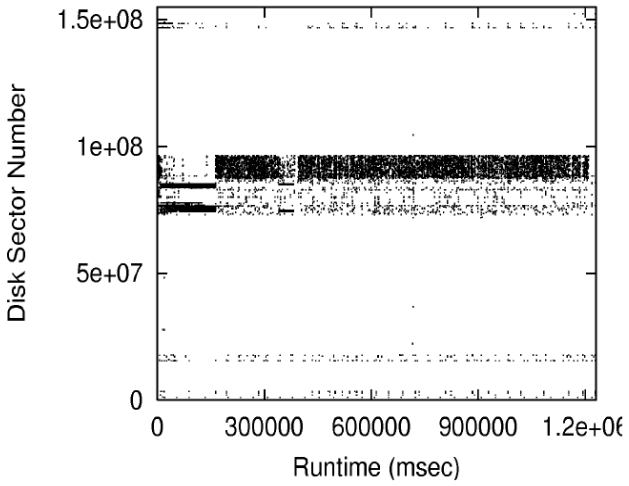


$$T_A = T_s + T_r + T_t \qquad T_s = \begin{cases} 8\text{E-}21x^3 - 2\text{E-}13x^2 + 1\text{E-}6x + 1.35 & x < 1.1 \times 10^7 \\ 9\text{E-}8x + 4.16 & \text{otherwise.} \end{cases}$$

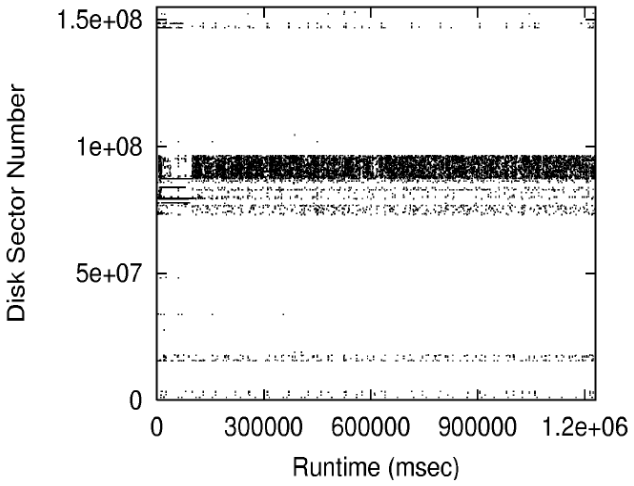
TPC-W Benchmark



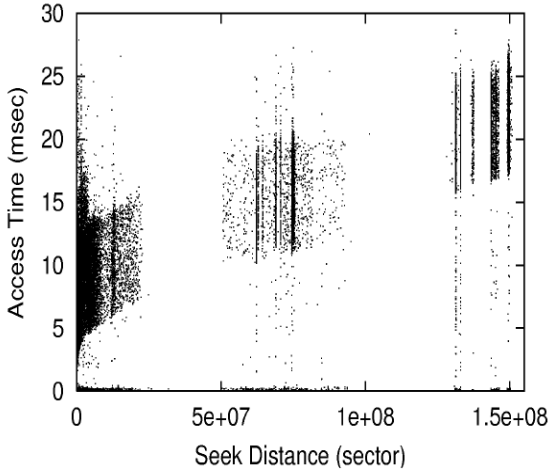
(a1) Ext2



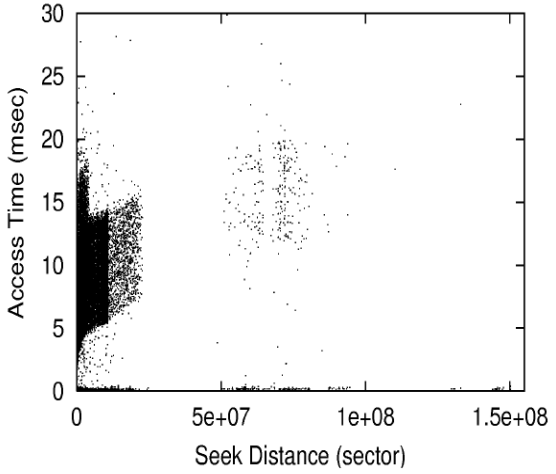
(b1) FS^2-static



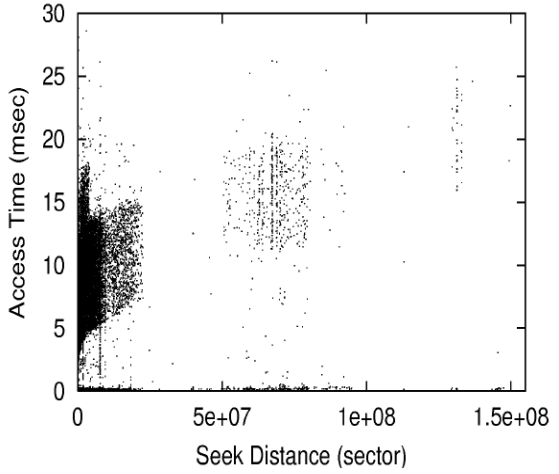
(c1) FS^2-dynamic



(a2) Ext2

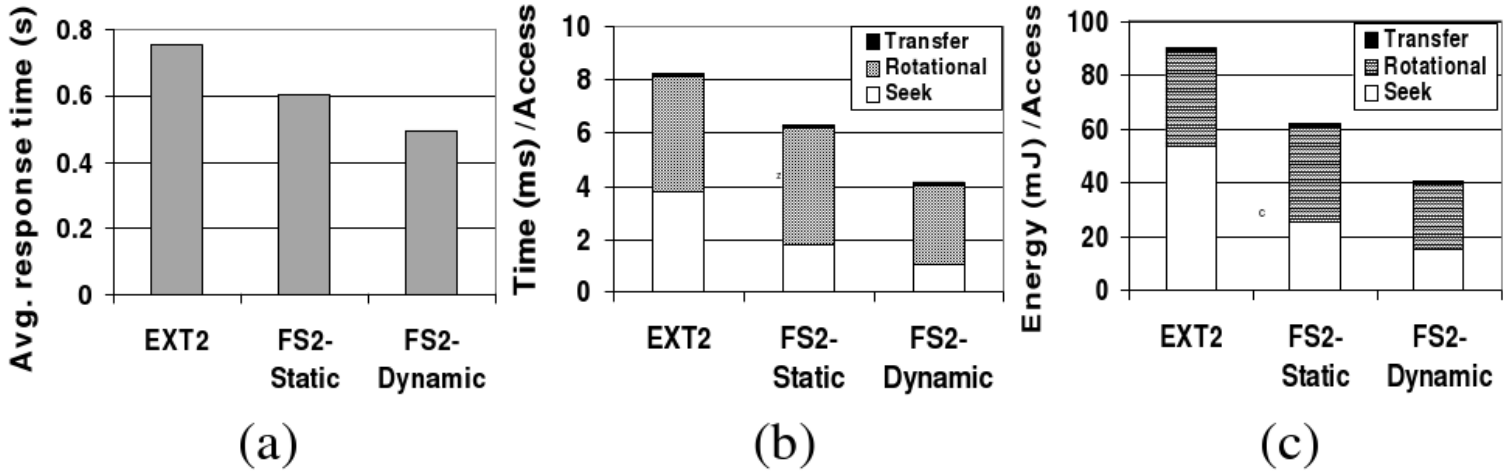


(b2) FS^2-static



(c2) FS^2-dynamic

TPC-W Benchmark (II)



	Disk Busy	Performance Improvement			Energy Improvement
		T_A	T_s	T_r	
FS^2-static	23%	24%	53%	-1.6%	31%
FS^2-dynamic	17%	50%	72%	31%	55%

FS^2 Static vs. FS^2 Dynamic

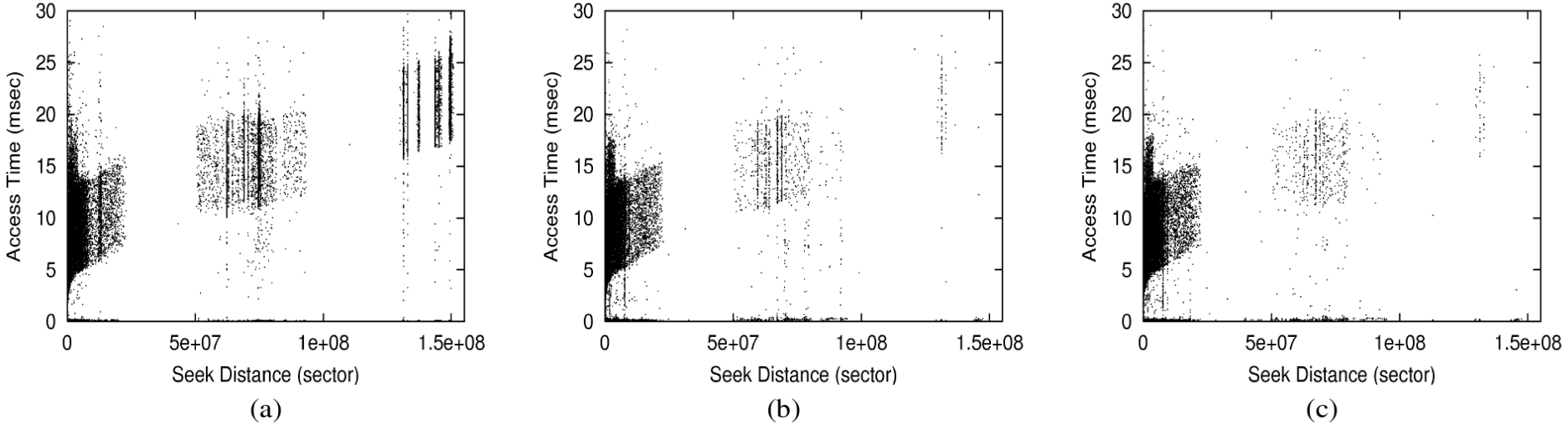
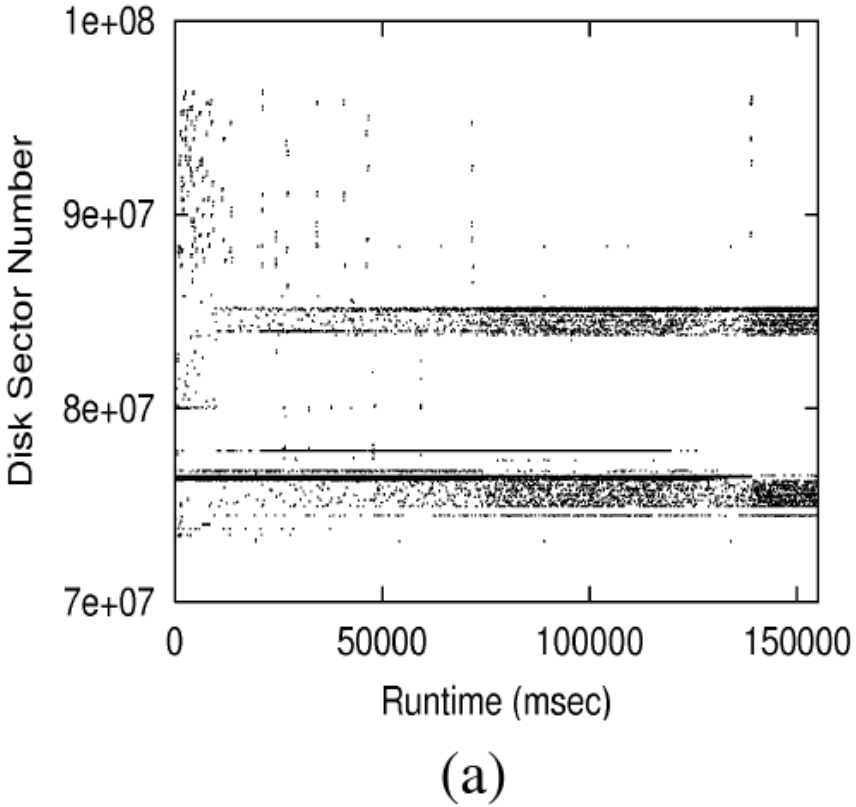
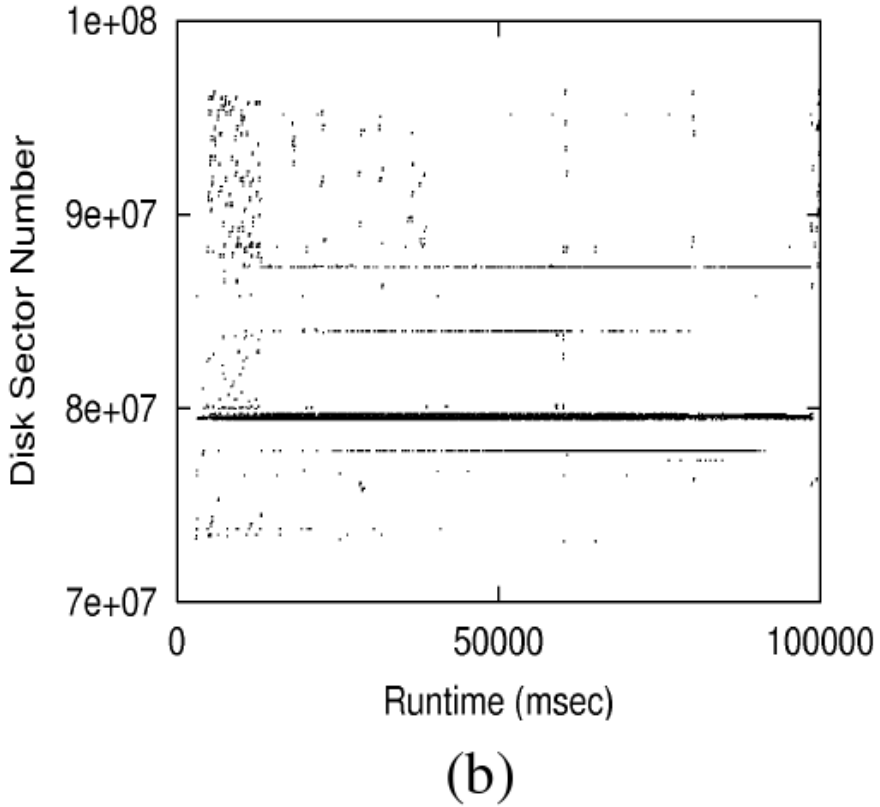


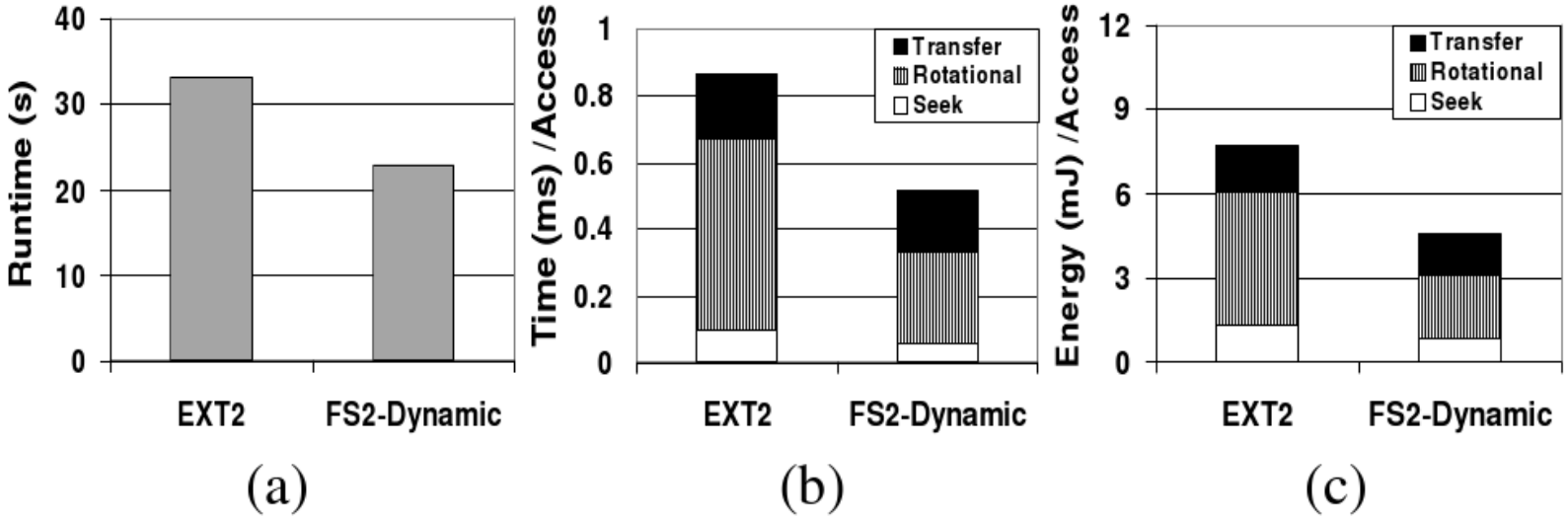
Figure 11: For the TPC-W benchmark, parts (a–c) show disk access time for the 1^{st}, the 2^{nd}, and the 7^{th} FS^2-dynamic run.

FS^2 Static vs. FS^2 Dynamic (II)



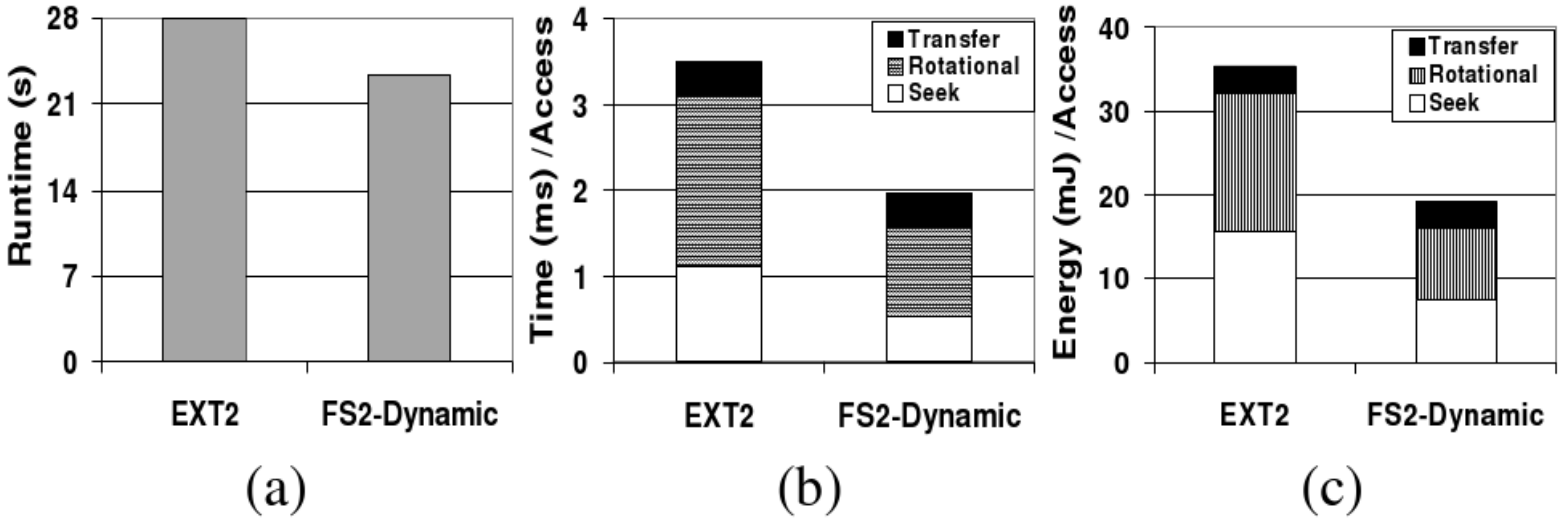


CVS Update Benchmark



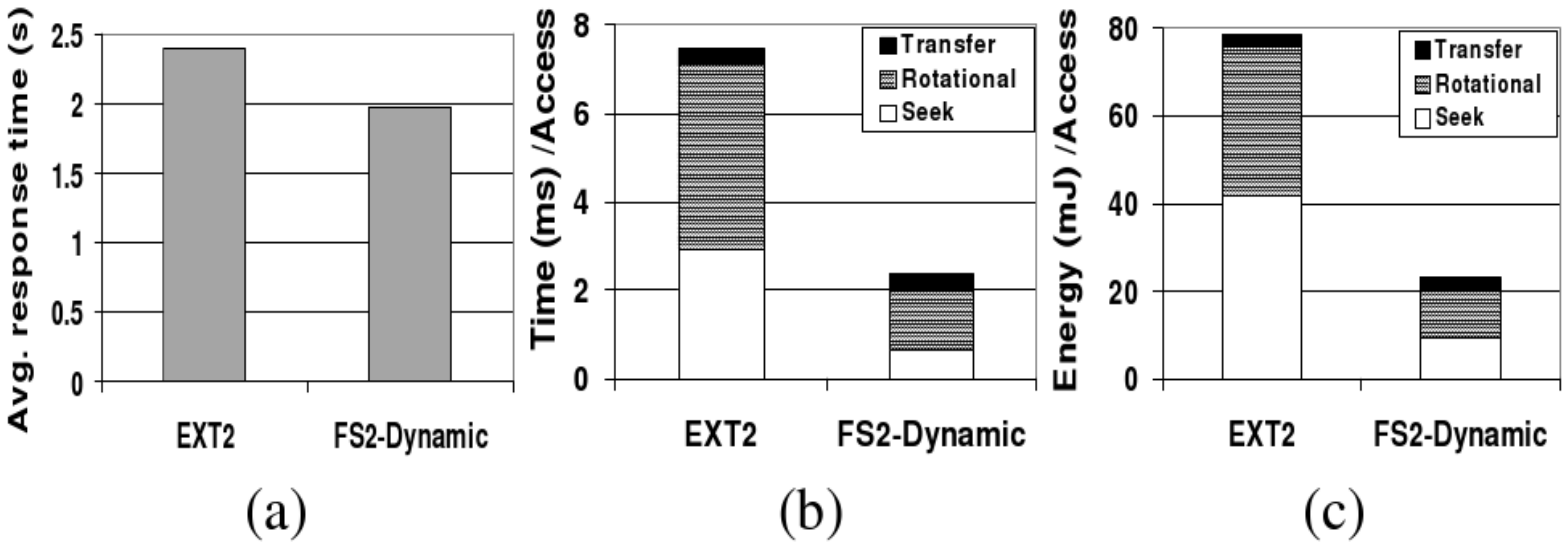
	Disk Busy	Performance Improvement			Energy Improvement
		T_A	T_s	T_r	
FS^2	73%	41%	38%	53%	40%

X Server + KDE Benchmark



	Disk Busy	Performance Improvement			Energy Improvement
		T_A	T_s	T_r	
FS^2	26%	44%	53%	47%	46%

SURGE Benchmark



	Disk Busy	Performance Improvement			Energy Improvement
		T_A	T_s	T_r	
FS^2	4.2%	69%	78%	68%	71%

Summary

- Make data placement on disk dynamic based on observed workload
- Replicate data to reduce disk access times (41-68% improvement)
- User-perceivable performance improvements for relevant workloads (16-34%)
- Lower energy consumption due to less seeking / rotational delay (40-71%)

Discussion Points

- Large improvements, but: "`cvs update`" still slow and lot of seeking activity
- What do we need?
  - Asynchronous I/O in applications to allow I/O scheduler to optimize globally?
  - I/O priorities? Hint low-latency vs. bulk I/O?
  - More aggressive read ahead?
  - Flash?
- Energy calculations accurate? What about increased CPU load?
- ZFS "ditto mode"?

References

- Hai Huang, Wanda Hung, Kang G. Shin, **"FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption"**, SOSP '05, Brighton, UK