The Rio File Cache:

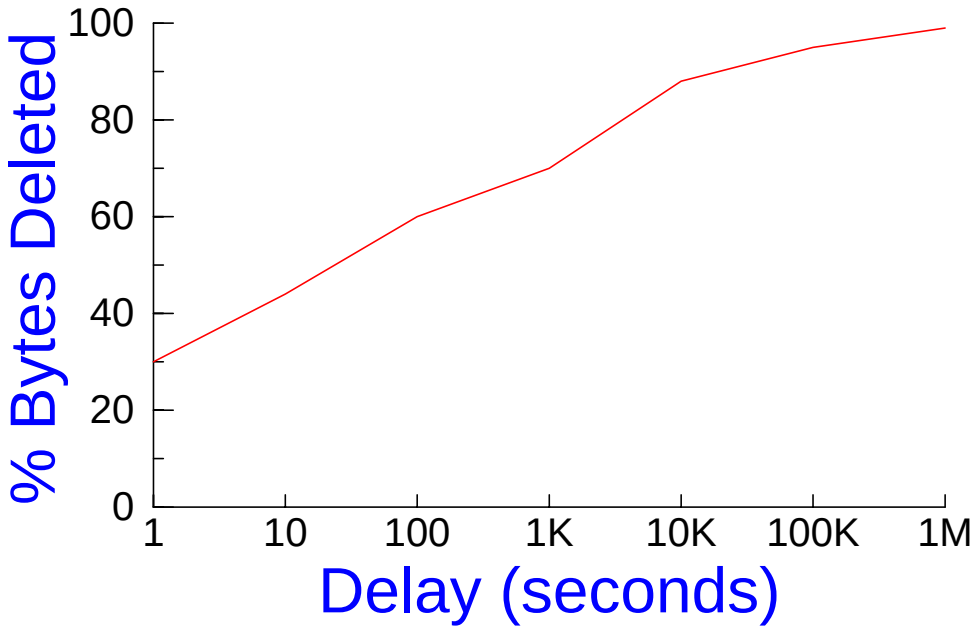
Surviving Operating System Crashes

Peter M. Chen, Wee Teck Ng,
Subhachandra Chandra, Christopher Aycock,
Gurushankar Rajamani, David Lowell

Computer Science and Engineering Division
Electrical Engineering and Computer Science
University of Michigan

Problem: Memory Unsafe for Files

Limits effectiveness of file cache [Baker91]



Forces trade-off between reliability and performance

write-through **delayed write** **write-back**

best reliability e.g. TP

best performance e.g. mfs

What Makes Memory Unreliable?

Vulnerable to power loss

fix with UPS (battery) or Flash RAM

Vulnerable to software errors/system crashes

"*Ordinary RAM memory...is wiped out when...a machine crashes*" [Tanenbaum95]

"*Volatile storage does not usually survive system crashes*" [Silberschatz94]

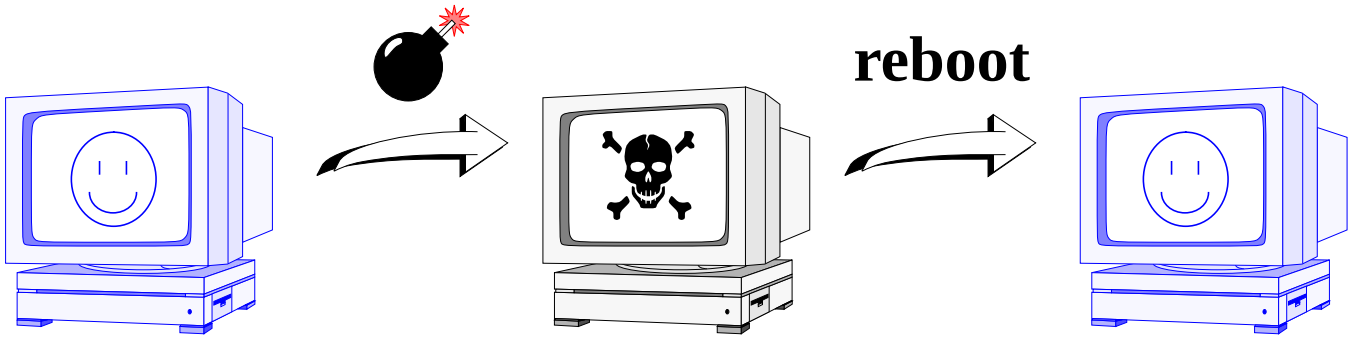
"*...safe from some software errors that write over main memory*" [Hennessy/Patterson96]

Disk vs. memory interface

writes to disk must go through complex protocol, error checking

any kernel store can write files in memory

Rio: A Reliable File Cache



protect memory
during crash

restore memory
during reboot

Remove reliability-induced writes to disk (sync, fsync, bwrite, bawrite)

Platform: DEC Alpha (128 MB memory), Digital Unix V3.0

Protect Memory by Controlling Access

Start with VM's write protection

- file cache pages normally kept write-protected
- unauthorized writes trigger exception
- file cache routines unprotect page before writing, reprotect and verify page after writing

But *physical addresses* can bypass VM

Force all addresses through VM

- extra address map translates physical addresses
- disable writes to file cache pages
- low overhead

Warm Reboot

Assuming file cache is preserved during crash, how to easily restore during reboot?

What additional data to maintain before crash?

- could use existing kernel data structures, but must protect each structure
- registry of essential info: physical memory address, device number, disk address, inode, file offset

Sync data/metadata to disk when reboot begins

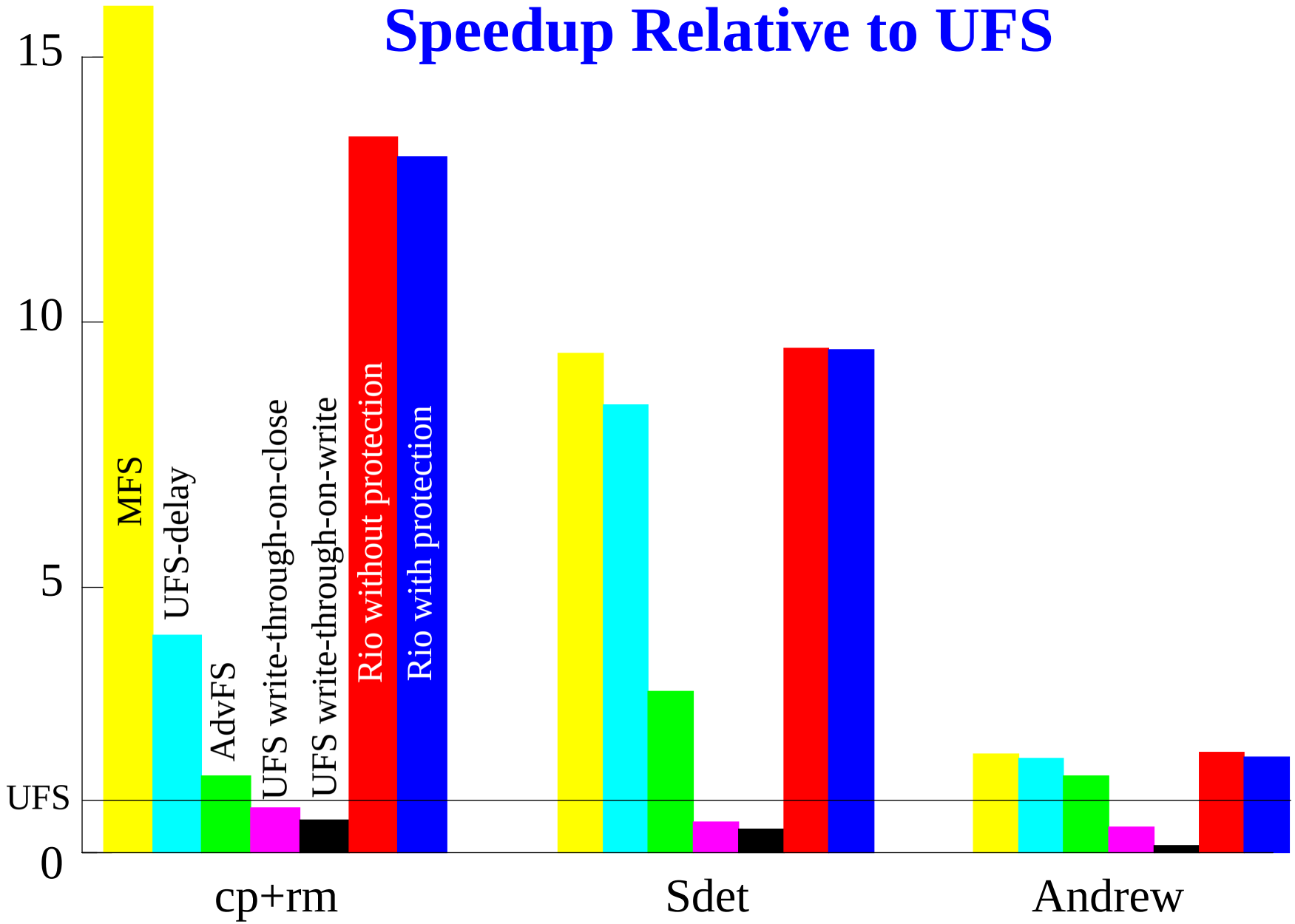
Crashing the Operating System

Fault models: goal is wide variety and realism

Fault Type	Example of Programming Error
destination register	numFreePages = count(freePageHeadPtr)
source register	numPages =physicalMemorySize/pageSize
delete branch	if while (...) {body}
delete random inst.	for (i=0; i<10; i++,j++) {body}
initialization	function () {int i=0; ...}
pointer	ptr = ptr->next->next;
allocation	ptr = malloc(); use ptr; use ptr; free(ptr);
copy overrun	for (i=0; i<sizeUsed; i++) {a[i] = b[i]};
off-by-one	for (i=0; i < <= size; i++)
synchronization	getWriteLock; write(); freeWriteLock;

Reliability Results

Fault Type	Disk-Based	Rio without Protection	Rio with Protection
kernel text	2	1	
kernel heap			
kernel stack		1	1
destination register			
source register	2		
delete branch	1	1	1
delete random inst.	1		
initialization			1
pointer		1	
allocation			
copy overrun		4	
off-by-one	1	2	1
synchronization			
Total	**7 of 650 (1.1%)**	**10 of 650 (1.5%)**	**4 of 650 (0.6%)**
MTTF With Crash Every 2 Months	**15 years**	**11 years**	**27 years**


Speedup Relative to UFS
15
10
5
UFS
0
MFS
UFS-delay
AdvFS
UFS write-through-on-close
UFS write-through-on-write
Rio without protection
Rio with protection
cp+rm
Sdet
Andrew

Conclusions

Rio makes memory as safe as disk

Benefits

reliability: all writes immediately and synchronously permanent

performance: no reliability-induced writes to disk

rio.eecs.umich.edu

running Rio file system with protection

stores our home directories, Rio source tree, mail