Journaling versus Softupdates

Asynchronous Meta-Data Protection in File System

Authors - Margo Seltzer, Gregory Ganger et all

Presenter – Abhishek Abhyankar
MS Computer Science
Virginia Tech

Overview of the Problem

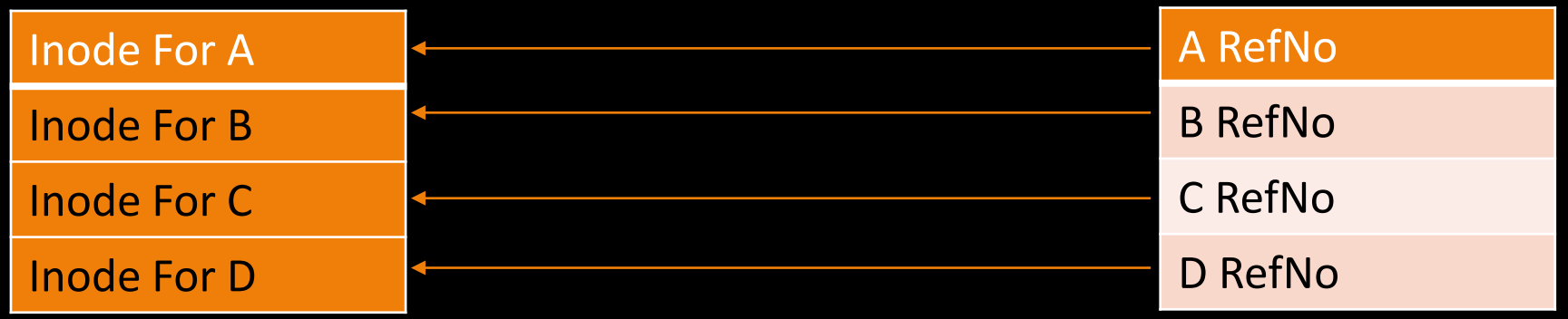
Metadata operations

- Create, Delete, Rename.
- Meta Data operations Modify the structure of the File System.

File System Integrity

- After a system crash, File System should be *recoverable to a consistent state* where it can continue to operate.

How is Integrity Compromised ?

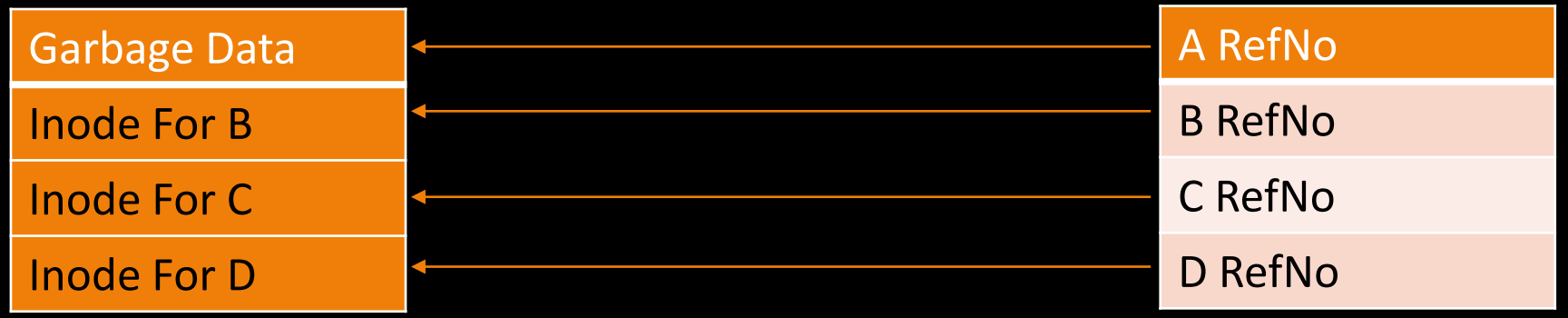


I-Node Block

Directory Block

- Suppose File A is Deleted.
- And First Node A is Deleted and Persisted to Disk.
- System Crash.

How is Integrity Compromised ?

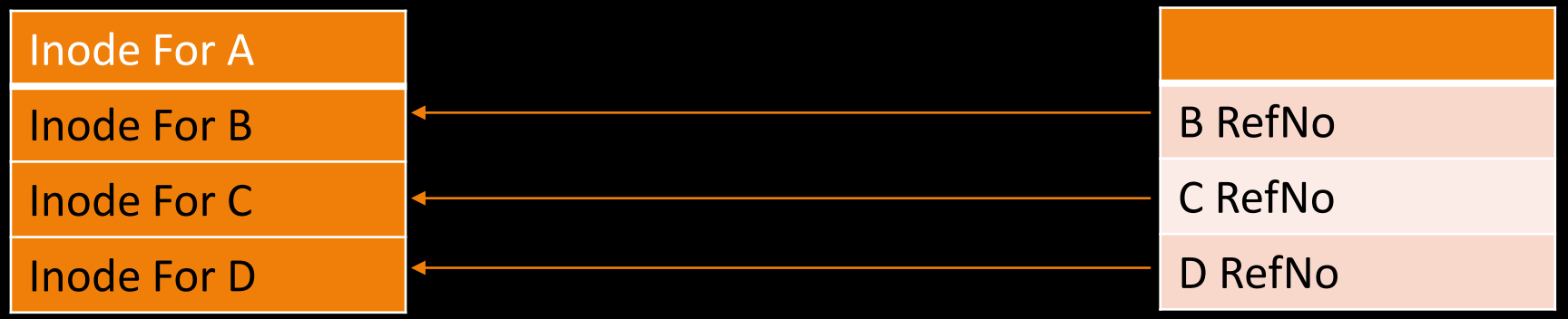


I-Node Block

Directory Block

- Garbage Data is present in the File A location.
- Directory reference is still pointing to the Garbage data,
- Integrity is compromised as there is no way to recover.

How Integrity can be Preserved?



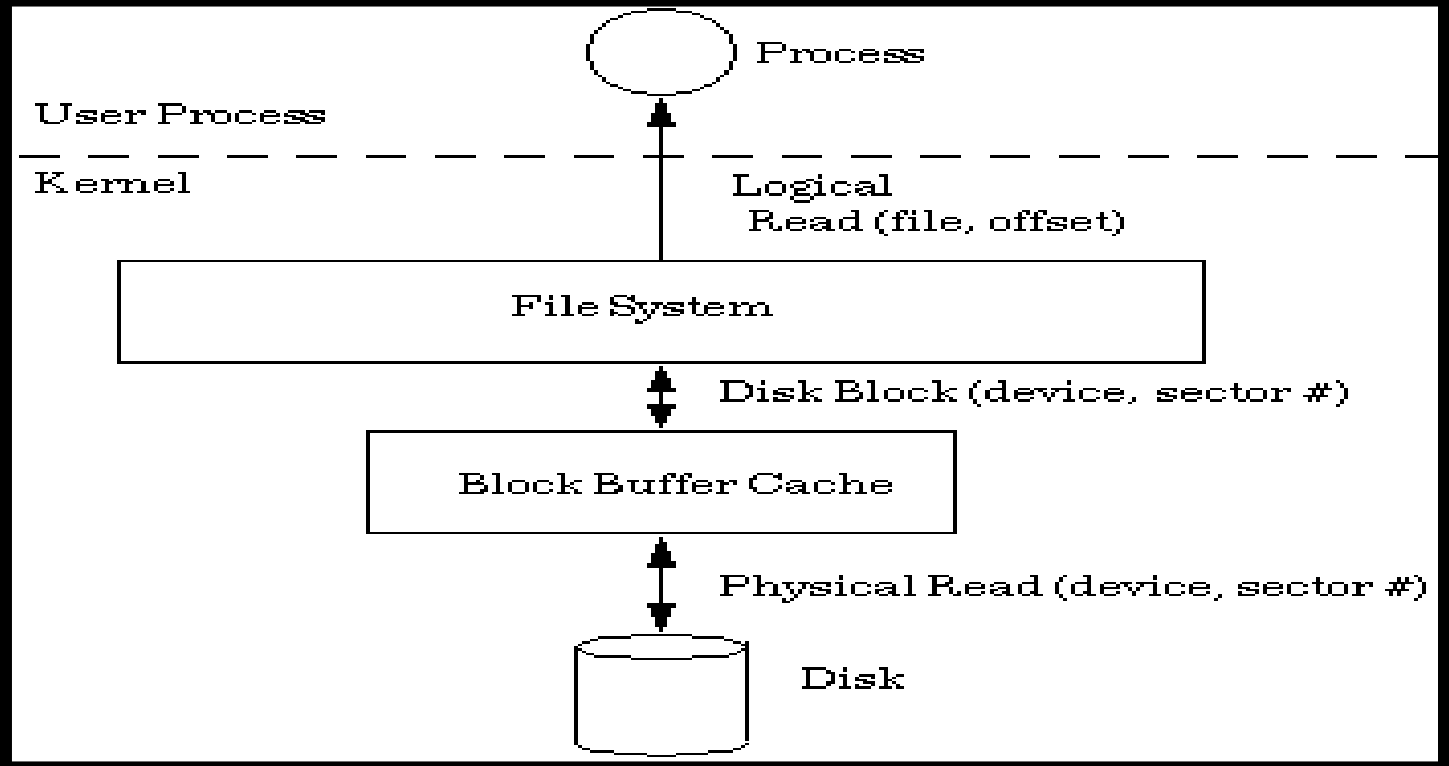
I-Node Block

Directory Block

- Directory reference is first deleted.
- System Crash.
- Orphan is created but Integrity is preserved.

What makes it difficult to handle?

- Multiple blocks are involved in a single logical operation
- Most update operations are asynchronous/delayed
- Actual IO ordering is done by Disk scheduler



Ordering Constraints

Deleting a file

- Delete the Directory entry
- Delete the I-node
- Delete Data Blocks

Creating a file

- Allocate the data blocks
- Allocate I-node
- Create Directory Entry

Solution:

Enforce the ordering constraints, *synchronously.*

Before the system call returns; the related metadata blocks are written synchronously in a correct order

Unix Fast File System with Synchronous Meta Data Updates.

> *BSD "synchronous" filesystem updates are braindamaged.*
>
> *BSD people touting it as a feature are WRONG. It's a bug.*
>
> *Synchronous meta-data updates are STUPID.*
>
> *… Linus Tovalds, 1995*
>
> *- Chief Architect and Project Coordinator Linux Kernel*

Asynchronous Updates

Disk access takes much more amount of time than the processor takes.

So why wait for the disk ?

Store the updates and return the system call and let the process continue.

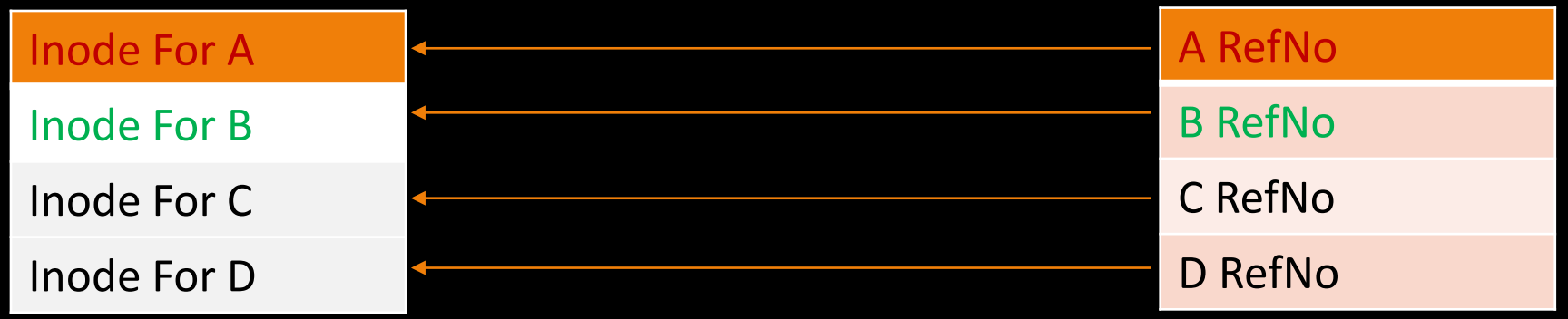
Perform Delayed writes to the disk.

Just maintain the ordering constraints which were mentioned earlier.

Soft Updates

- Enforce the ordering constraints, in an *asynchronously* way.
- Maintain dirty blocks and dependencies to each other.
- Let Disk Scheduler sync any disk blocks.
- When a block is written by Disk Scheduler, Soft Update code can take care of the dependencies.
- Maintains the Dependency information on Pointer basis not Block basis.

Cyclic Dependencies

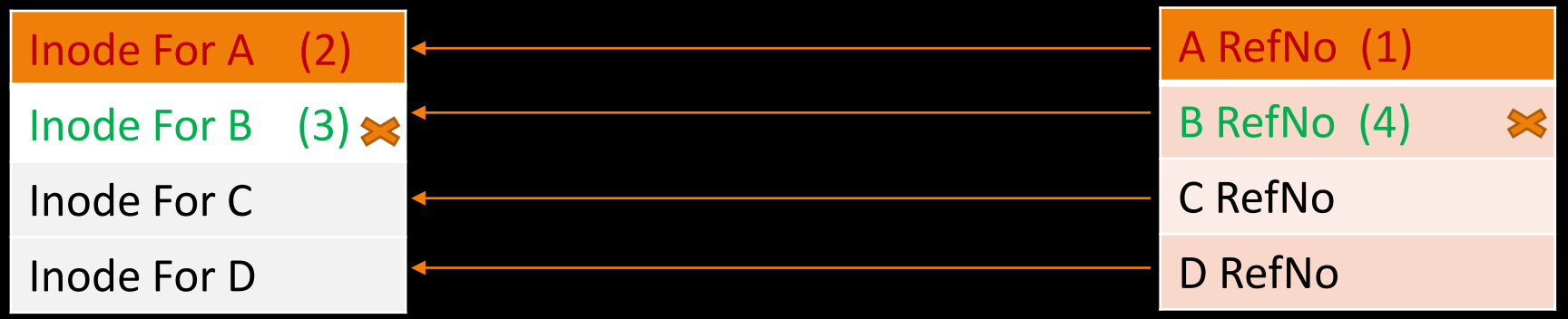


I-Node Block

Directory Block

- File A is Created.
- File B is Deleted.
- Node A needs to be created before Dir A is created.
- Dir B needs to be removed before Node is removed.

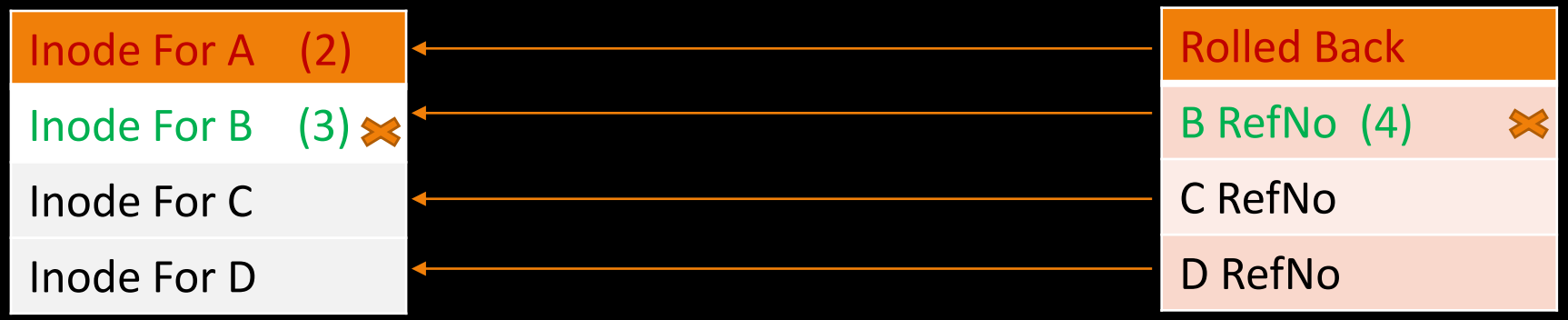
How is Dependency Resolved ?



I-Node Block

Directory Block

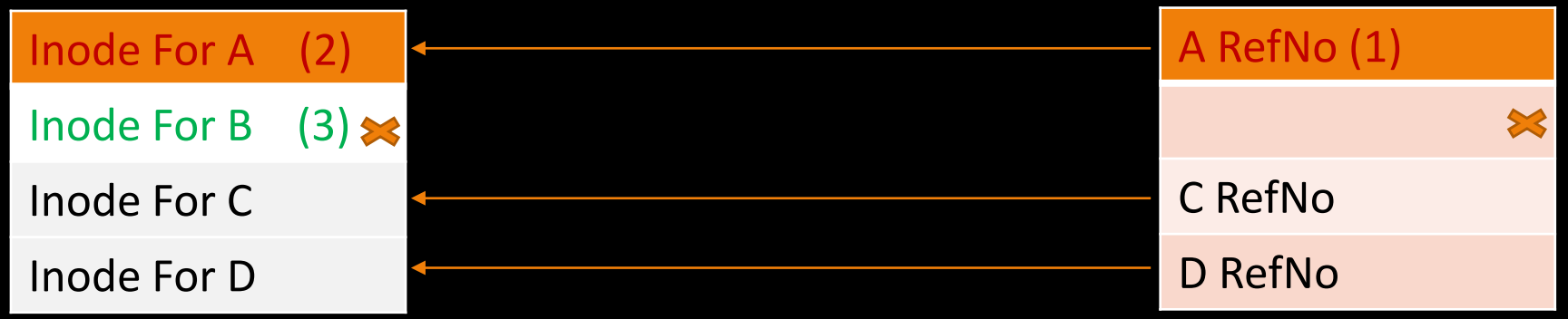
- File A is Created. (1) Depends On (2)
- File B is Deleted. (3) Depends On (4)
- Disk Scheduler selects Directory Block and notifies Soft Update.



I-Node Block

Directory Block

- As (1) Depends On (2). (1) is rolled back to original state.
- As (4) does not depend on anyone, it is executed i.e removed.
- Dependency (3) Depends on (4) is removed.

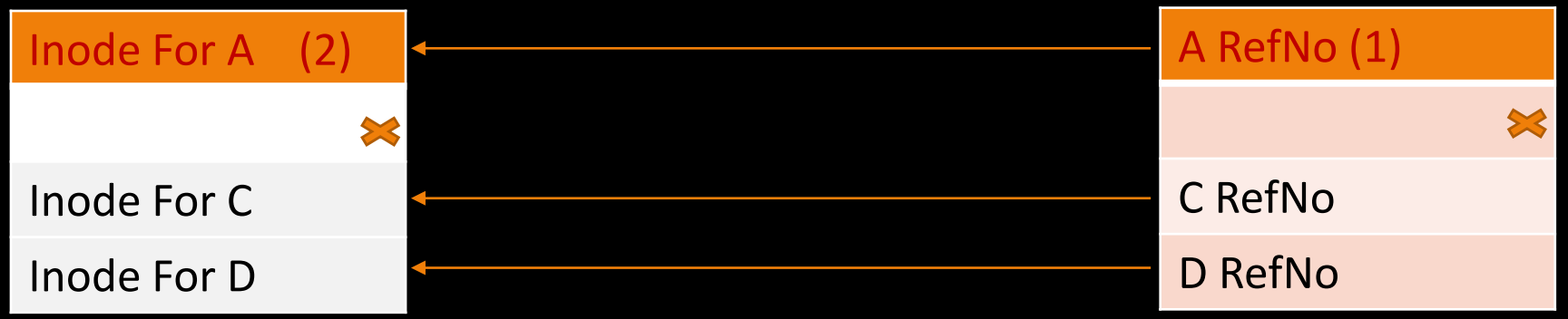


I-Node Block

Directory Block

- Now after Directory block is persisted. Inode Block is selected. (Dir A is Rolled forwarded again).
- (2) and (3) are executed. i.e (2) is created and (3) is removed.
- Then Dir block is selected again and executes (1).

Returned to Stable State



I-Node Block

Directory Block

- After a sequence of instructions all dependencies are resolved and the system returns to stable state.
- Even if system crashed anywhere in the middle File system integrity will always be maintained.

Soft Updates Conclusion

Advantages:

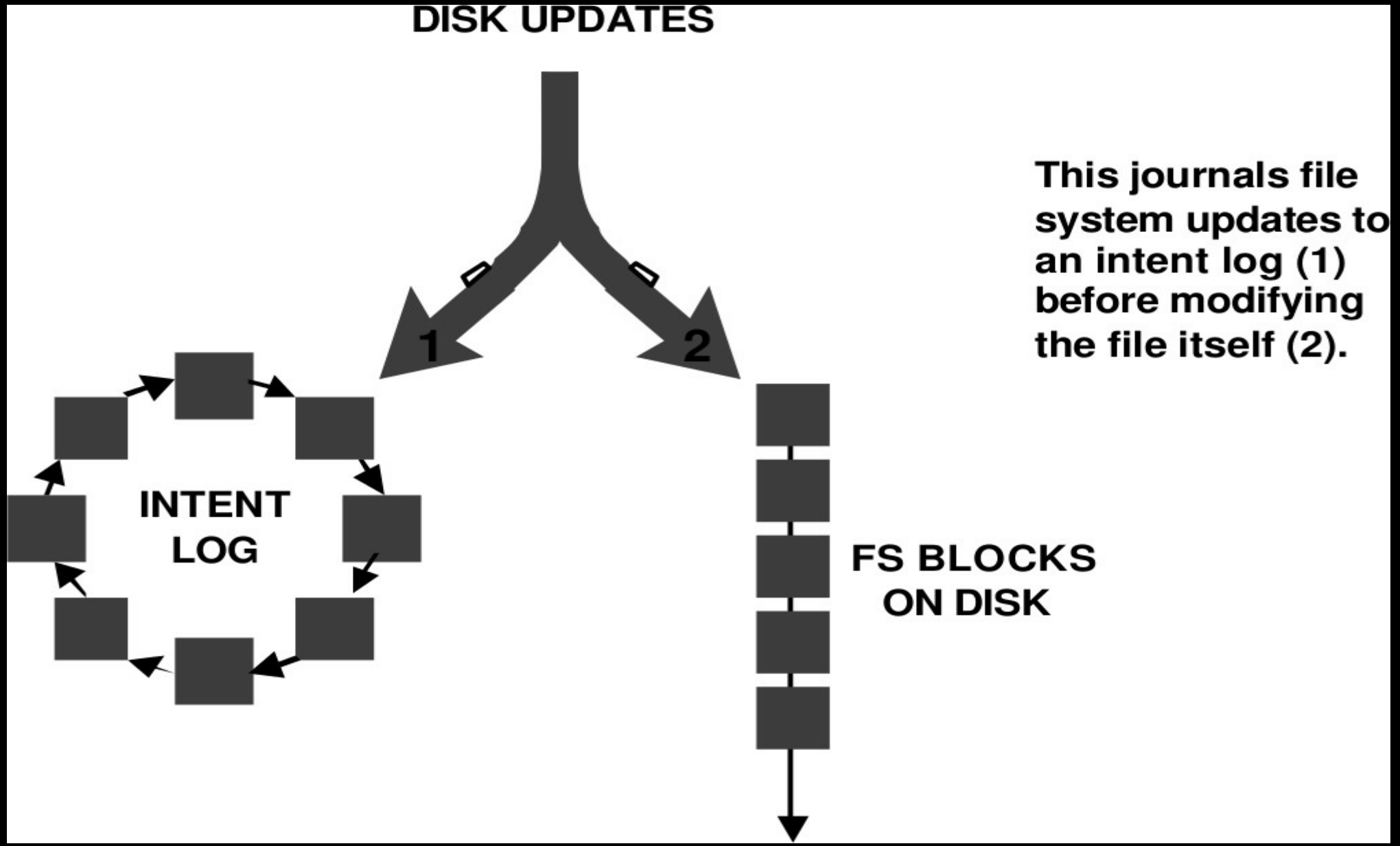
- No recovery required. Directly mount and play.
- Still enjoys delayed writes.

Disadvantages:

- Orphan nodes might get created.
- Integrity guaranteed, but still background fsck is required.
- Implementation code is very complex.

Journaling

- Write ahead logging.
- Write changes to metadata in the journal.
- Blocks are written to disk *only after associated journal data has been committed.*
- On recovery, just replay for committed journal records.
- Guarantees Atomic Metadata operations.


DISK UPDATES
1
2
INTENT
LOG
FS BLOCKS
ON DISK
This journals file system updates to an intent log (1) before modifying the file itself (2).

Different Implementations of Journaling

LFFS-file

- Writes log records to a *file*
- Writes log records *asynchronously*
- 64KB cluster
- Each buffered cached block has relevant Log entry as Header and Footer

Different Implementations of Journaling

LFFS-wafs

- Writes log records to a separate filesystem
- Provides Flexibility.
- WAFS is minimal operations filesystem specially designed for Logging purpose.
- Uses LSN's (Low and High LSN).
- Complex than LFFS-File implementation

Recovery After a Crash

- First Log is recovered from the disk.
- The last log entry to disk is stored in the Superblock.
- That entry acts like a starting point. Any entries after that point will be validated and then either persisted or aborted.

Journaling Concluding Remarks

Advantages

- Quick recovery (fsck)

Disadvantages

- Extra IO generated

Parameters for Evaluation ?

FFS, FFS-async, LFFS-File, LFFS-WAFS, Softupdates are evaluated on these parameters.

- Durability of the Meta data Operations.
- Status of the file system after reboot.
- Guarantees provided of the data files after recovery.
- Atomicity.

Feature Comparison

Feature	File Systems
Meta-data updates are synchronous	FFS, LFFS-wafs-[12]sync
Meta-data updates are asynchronous	Soft Updates , LFFS-file, LFFS-wafs-[12]async
Meta-data updates are atomic.	LFFS-wafs-[12]* , LFFS-file
File data blocks are freed in background	Soft Updates
New data blocks are written before inode	Soft Updates
Recovery requires full file system scan	FFS
Recovery requires log replay	LFFS-*
Recovery is non-deterministic and may be impossible	FFS-async

Performance Measurement

Benchmarks

- Microbenchmark - only metadata operations (create/delete)

  Softupdates performs better in deletes but increased load, Journaling is better.

- Macrobenchmarks - real workloads

System Configurations:

	FreeBSD Platform
Motherboard	Intel ASUS P38F, 440BX Chipset
Processor	500 Mhz Xeon Pentium III
Memory	512 MB, 10 ns
Disk	3 9 GB 10,000 RPM Seagate Cheetahs Disk 1: Operating system, /usr, and swap Disk 2: 9,088 MB test partition Disk 2: 128 MB log partition Disk 3: 128 MB log partition
I/O Adapter	Adaptec AHA-2940UW SCSI
OS	FreeBSD-current (as of 1/26/00 10:30 PM) config: GENERIC + SOFTUPDATES – bpfilter – unnecessary devices

Table 3. System Configuration.

Micro benchmark Results

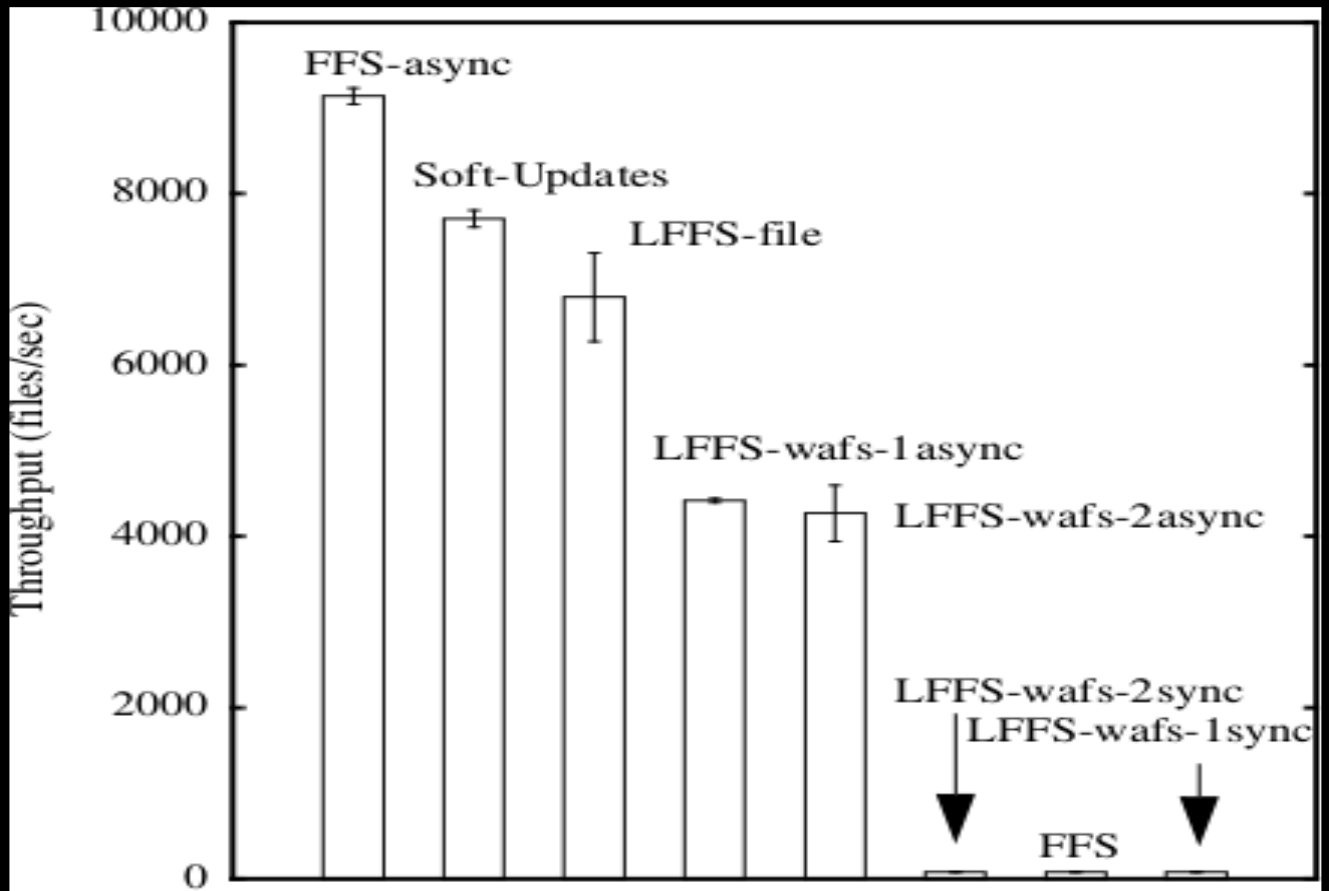


Figure 3. 0-length File Create/Delete Results in Files per Second.

Macro benchmarks workloads

- SSH. -> Unpack Compile and Build
- Netnews. -> Unbatch and Expire
- SDET.
- Post-Mark. -> Random Operations

Result Evaluation

- CPU intensive activities are almost identical across all filesystems.
- NetNews has heavy loads where Softupdates pays heavy penalty.
- SSH is Meta-data intensive so Softupdates performs better than all other filesystems.
- Postmarks demonstrates identical performance with Softupdates performing slighhtly better.

Macro benchmarks

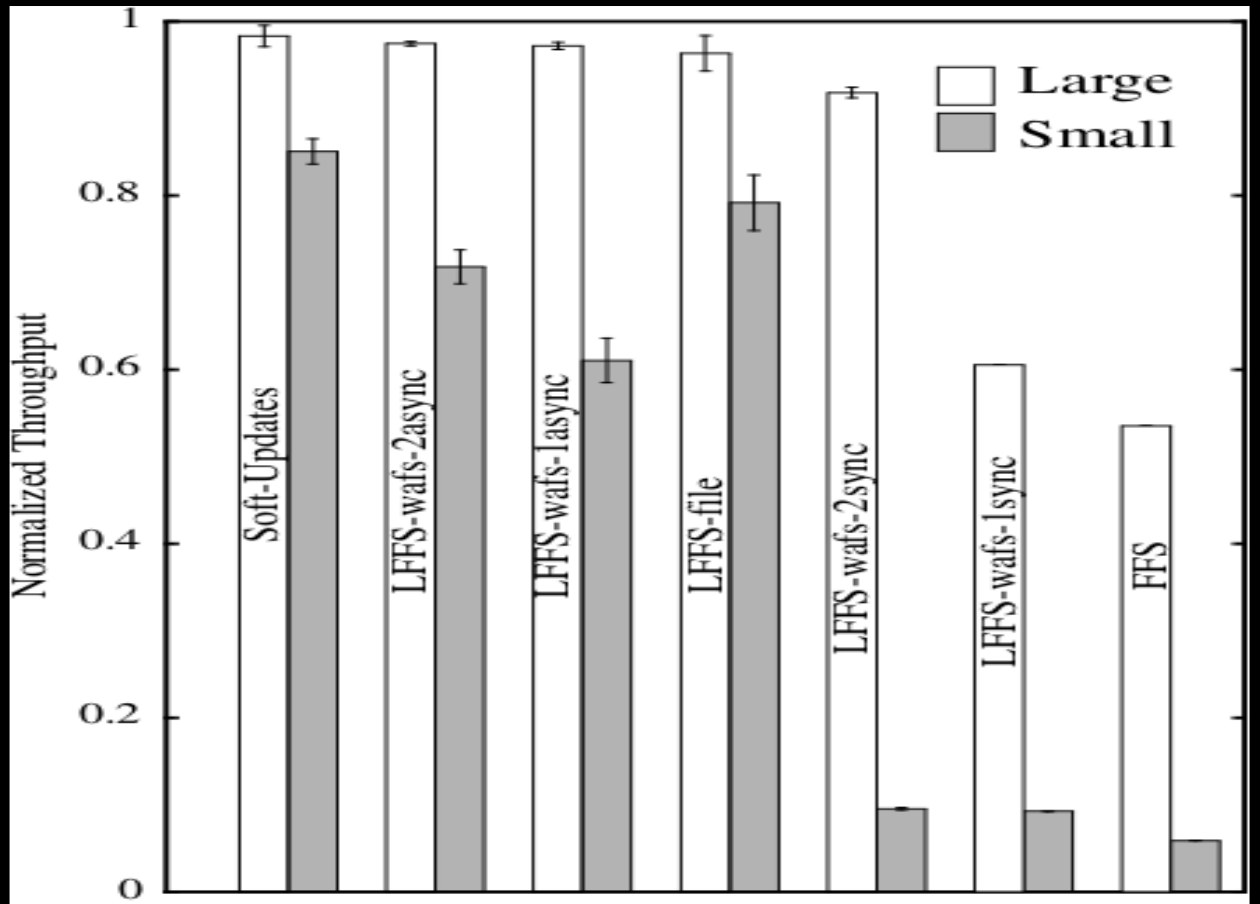


Figure 5. PostMark Results. The results are the averages of five runs; standard deviations are shown with error bars.

Concluding Remarks

- Displayed that Journaling and Soft Updates are both comparable at High Level.
- At lower level both provide a different set of useful semantics.
- Soft Updates performs better for Delete intensive workloads and small data sets.
- Assuming that Data sets are metadata intensive is unrealistic
- Journaling works fine with larger data sets and is still most widely used Filesystem Metadata recovery system.

Discussion ???

Thank You.

References

“Non-Volatile Memory for Fast, Reliable File Systems”

“Heuristic Cleaning Algorithms in Log-Structured File Systems”

“Journaling and Softupdates: Presentation Hyogi”

“The Rio File Cache: Surviving Operating System Crashes,”

“A Scalable News Architecture on a Single Spool,”

“The Episode File System,”

“Soft Updates: A Solution to the Metadata Update Problem in File Systems”

“Soft Updates: A Technique for Eliminating Most Synchronous Writes in the Fast Filesystem”

“The Write-Ahead File System: Integrating Kernel and Application Logging”