

Speculative Execution in a Distributed File System

Ed Nightingale

Peter Chen

Jason Flinn

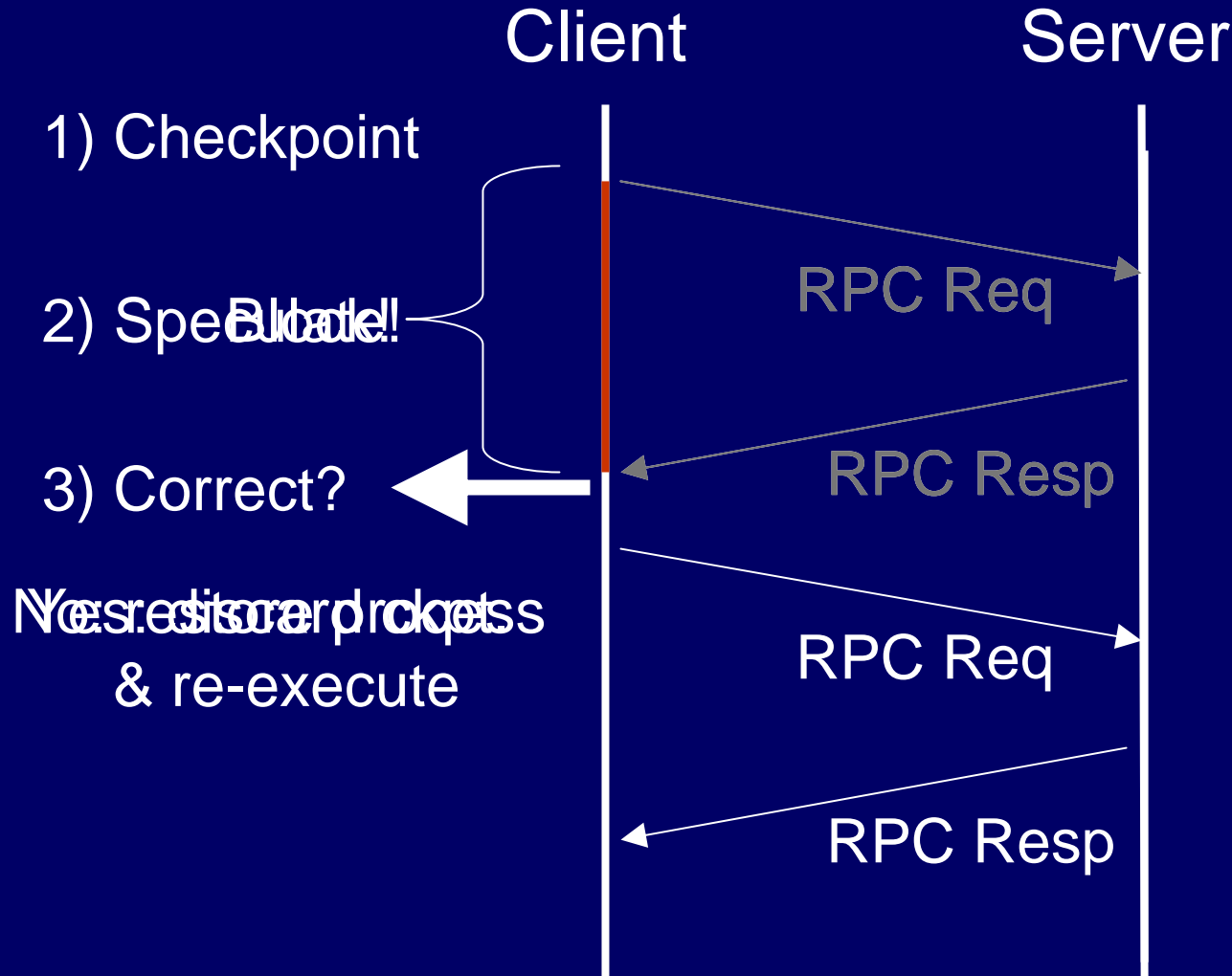
University of Michigan

Presented by Adam Boot

Motivation

- Why are distributed file systems slow(er)?
 - Sync b/w messages provide consistency
 - Sync disk writes provide safety
- Sacrifice guarantees for speed
- Can DFS can be safe, consistent and fast?
 - **Yes!** With OS support for speculative execution

Big Idea: Speculative



- Guarantees without blocking I/O!

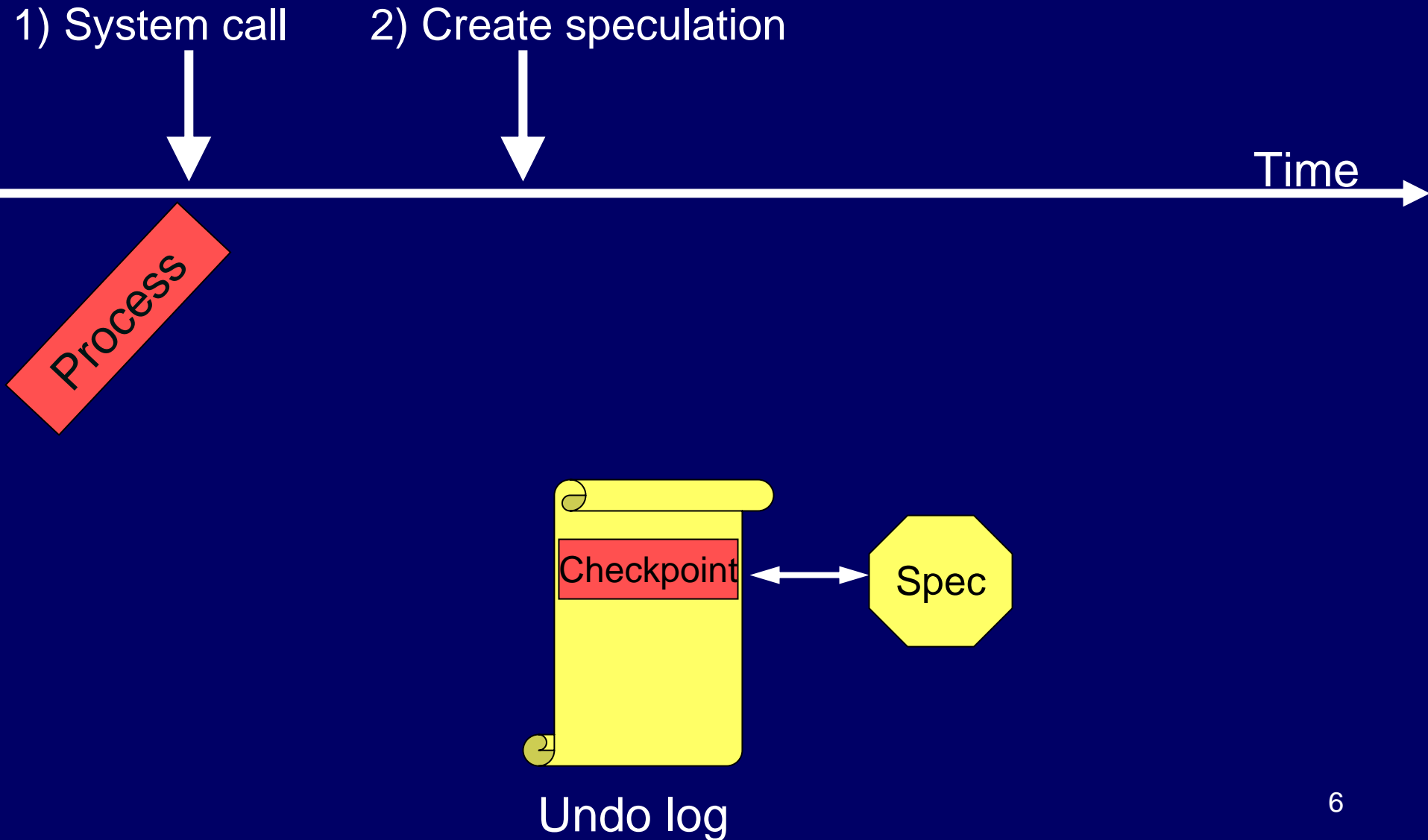
Conditions for Success

- Operations are highly predictable
 - Conflicts are rare
- Checkpoints are cheaper than network I/O
 - 52 μ s for small process
- Computers have resources to spare
 - Need memory and CPU cycles for speculation

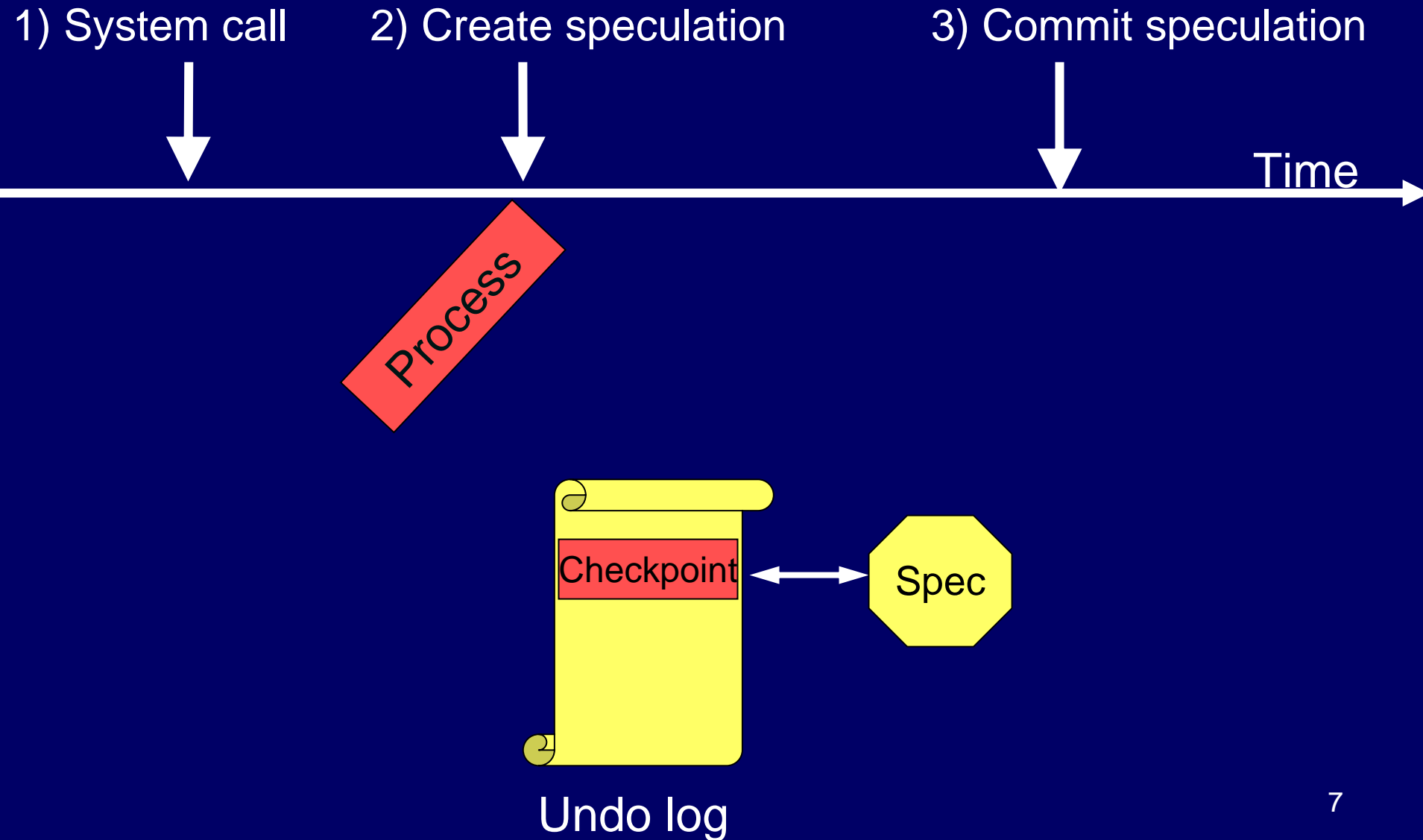
Outline

- Motivation
- **Implementing speculation**
- Multi-process speculation
- Using Speculator
- Evaluation

Implementing Speculation



Speculation Success

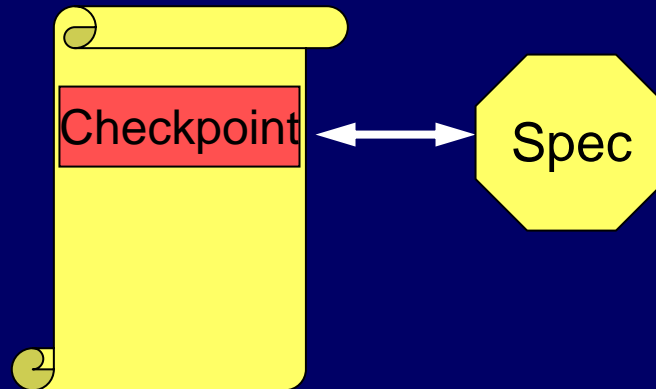
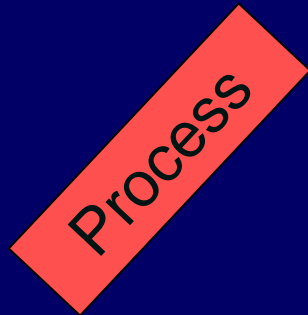
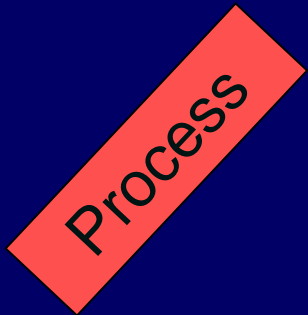


Speculation Failure

1) System call

2) Create speculation

3) Fail speculation



Undo log

Ensuring Correctness

- Spec processes often affect external state
- Three ways to ensure correct execution
 - Block
 - Buffer
 - Propagate speculations (dependencies)

Systems Calls

- Block calls that externalize state
 - Allow read-only calls (e.g. getpid)
 - Allow calls that modify only task state (e.g. dup2)
- File system calls -- need to dig deeper
 - Mark file systems that support Speculator

getpid	→	Call sys_getpid()
reboot	→	Block until specs resolved
mkdir	→	Allow only if fs supports Speculator

Output Commits

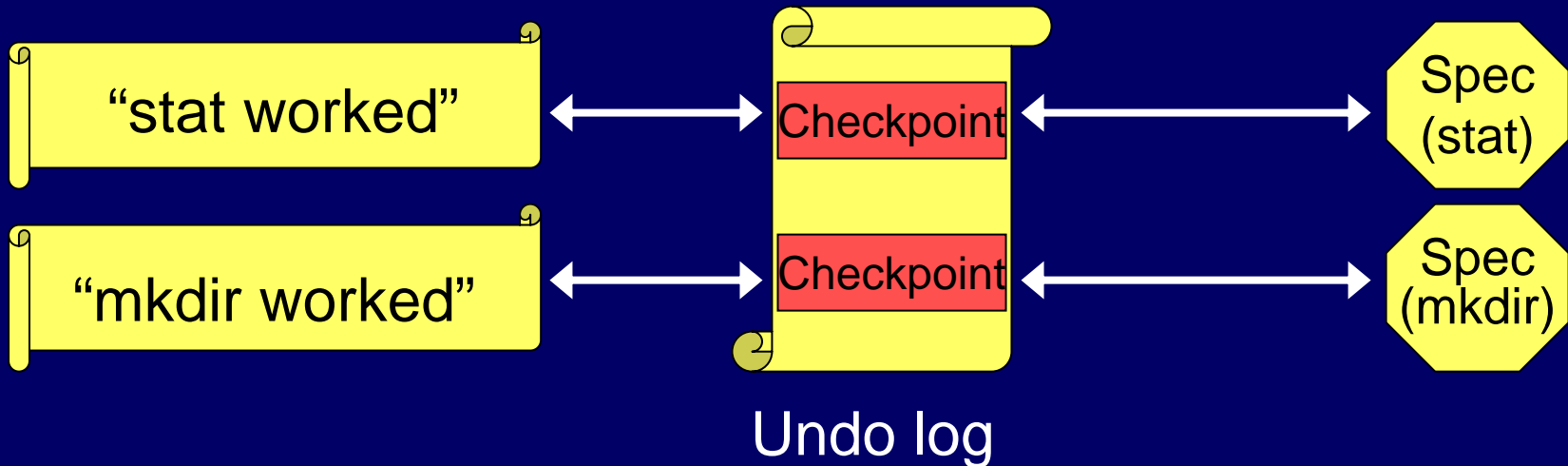
1) sys_stat

2) sys_mkdir

3) Commit speculation

Time

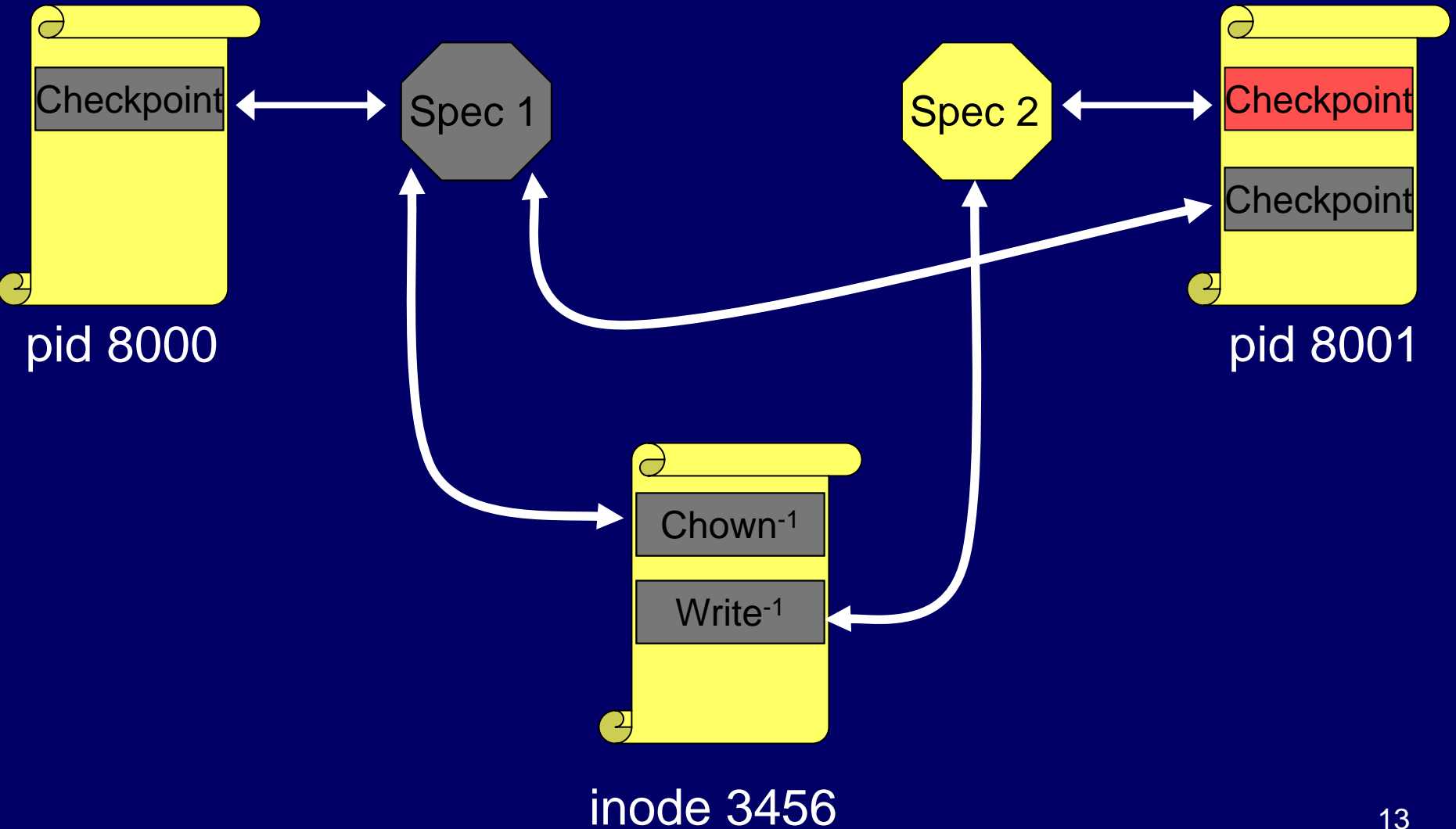
Process



Multi-Process Speculation

- Processes often cooperate
 - Example: “make” forks children to compile, link, etc.
 - Would block if speculation limited to one task
- Allow kernel objects to have speculative state
 - Examples: inodes, signals, pipes, Unix sockets, etc.
 - Propagate dependencies among objects
 - Objects rolled back to prior states when specs fail

Multi-Process Speculation



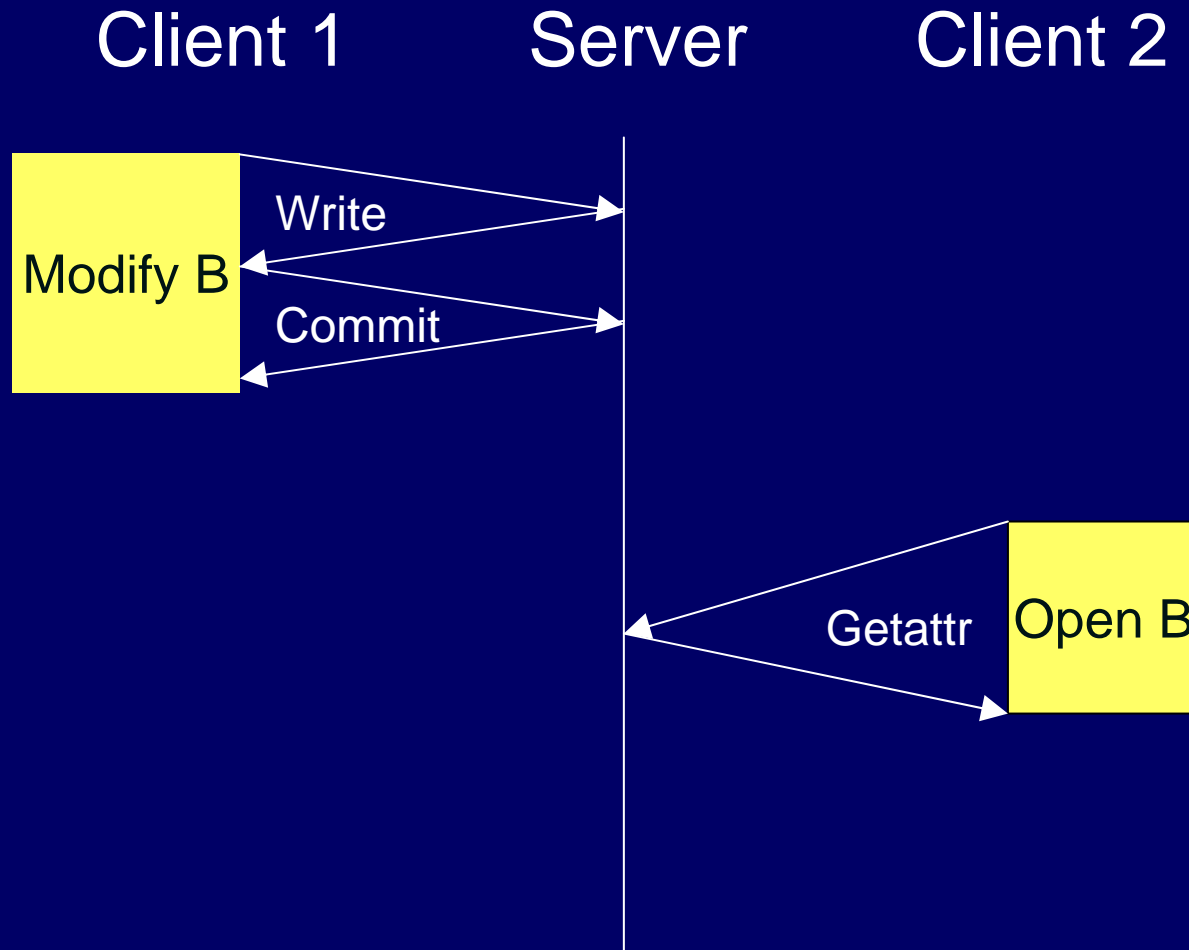
Multi-Process Speculation

- What we handle:
 - DFS objects, RAMFS, Ext3, Pipes & FIFOs
 - Unix Sockets, Signals, Fork & Exit
- What we don't (i.e. we block)
 - System V IPC
 - Multi-process write-shared memory

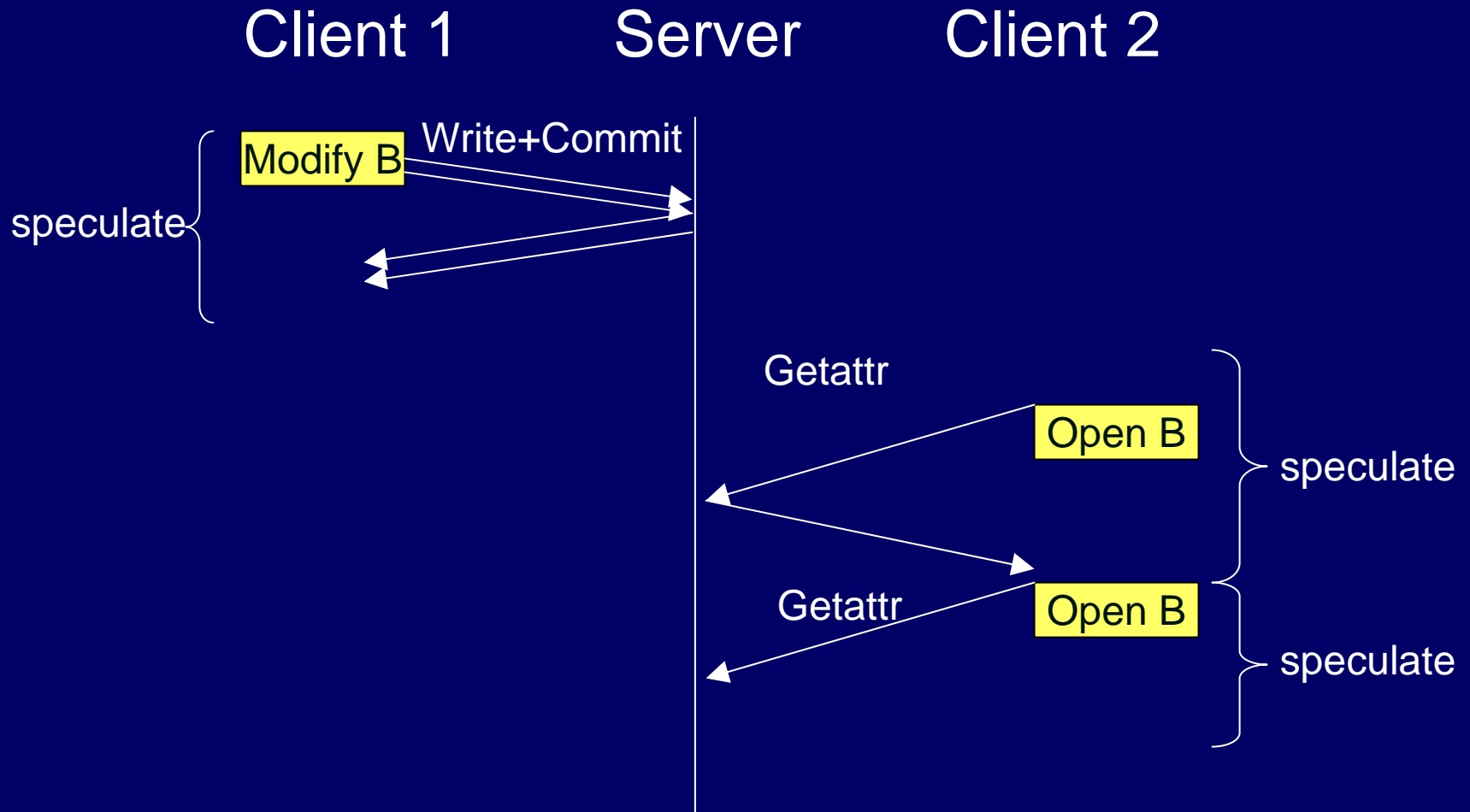
Outline

- Motivation
- Implementing speculation
- Multi-process speculation
- Using Speculator
- Evaluation

Example: NFSv3 Linux



Example: SpecNFS



Problem: Mutating Operations

Client 1

1. cat foo > bar

Client 2

2. cat bar

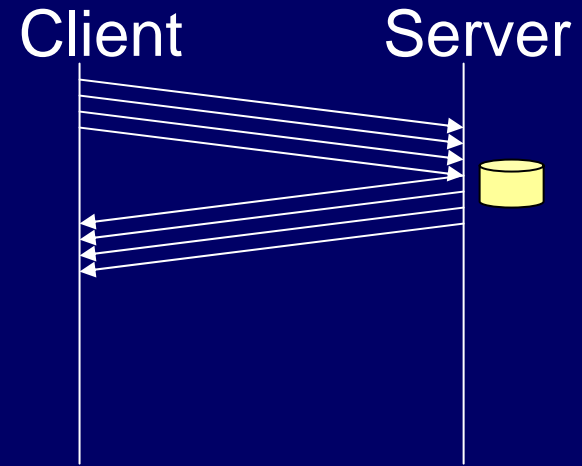
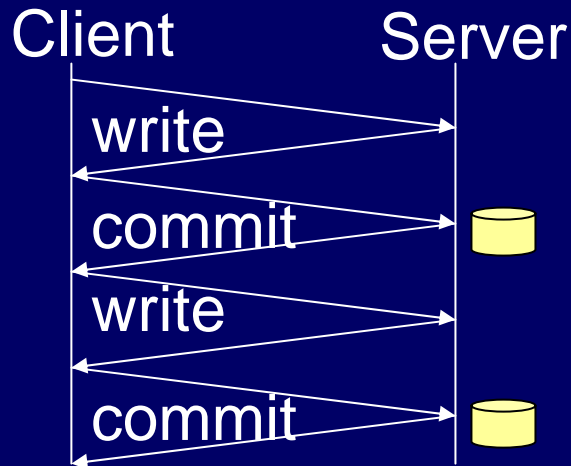
- bar depends on cat foo
- What does client 2 view in bar?

Solution: Mutating Operations

- Server determines speculation success/failure
 - State at server never speculative
- Send server hypothesis speculation based on
 - List of speculations an operation depends on
- Requires server to track failed speculations
- Requires in-order processing of messages

Group Commit

- Previously sequential ops now concurrent
- Sync ops usually committed to disk
- Speculator makes group commit possible



Putting it all Together: SpecNFS

- Apply Speculator to an existing file system
- Modified NFSv3 in Linux 2.4 kernel
 - Same RPCs issued (but many now asynchronous)
 - SpecNFS has same consistency, safety as NFS
 - Getattr, lookup, access speculate if data in cache
 - Create, mkdir, commit, etc. always speculate

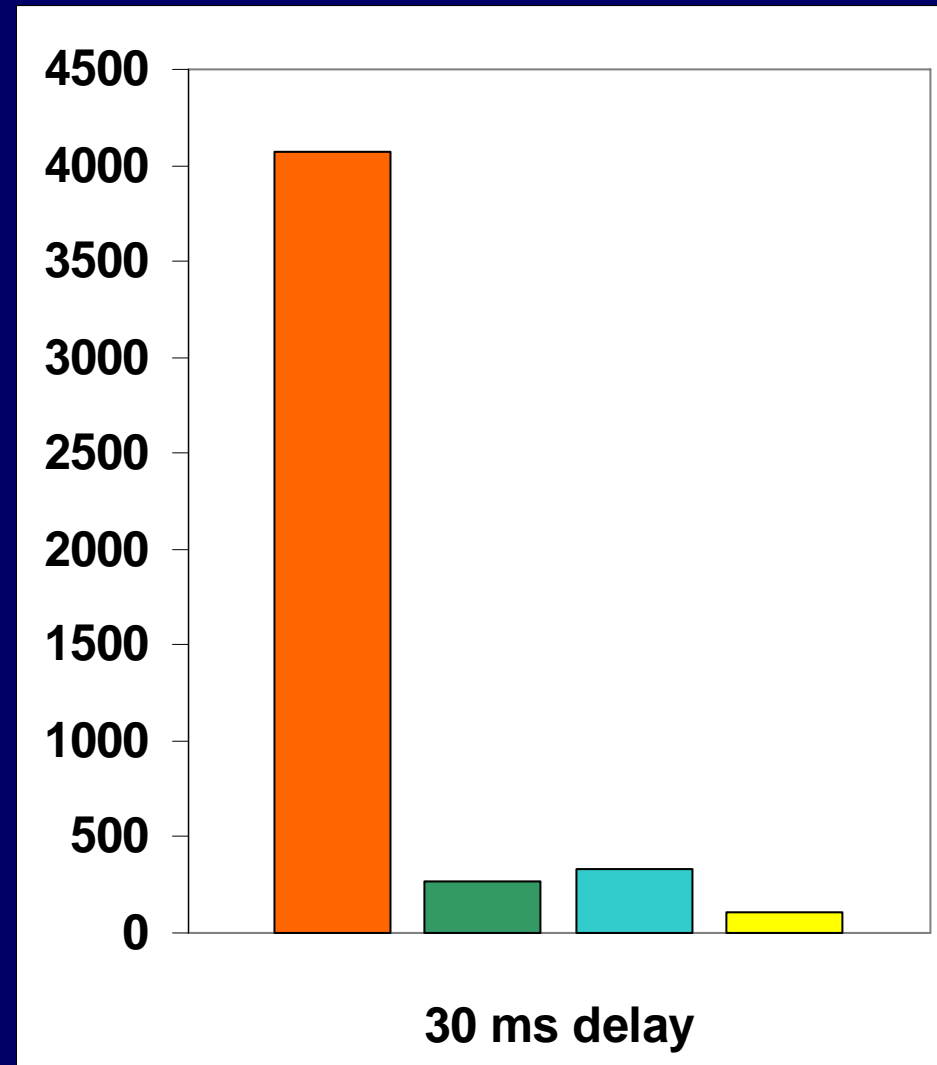
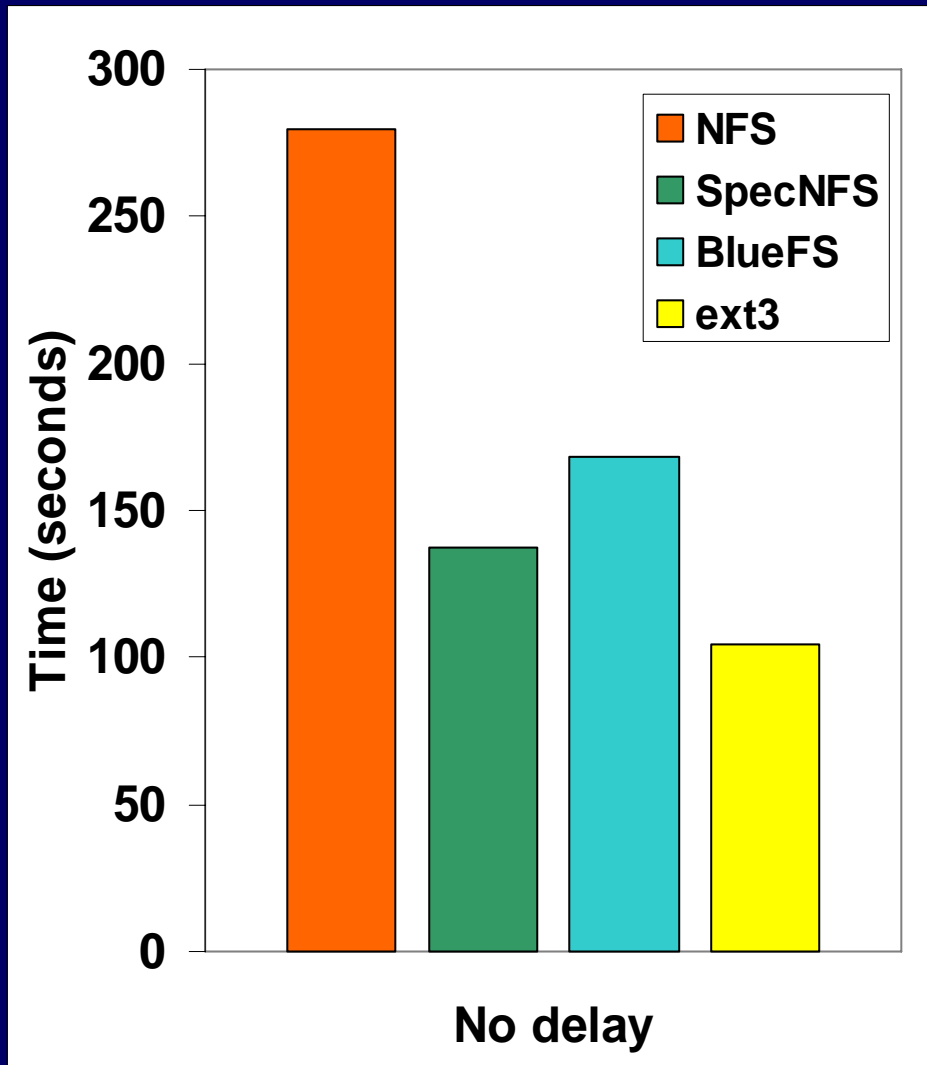
Putting it all Together: BlueFS

- Design a new file system for Speculator
 - Single copy semantics
 - Synchronous I/O
- Each file, directory, etc. has version number
 - Incremented on each mutating op (e.g. on write)
 - Checked prior to all operations.
 - Many ops speculate and check version async

Outline

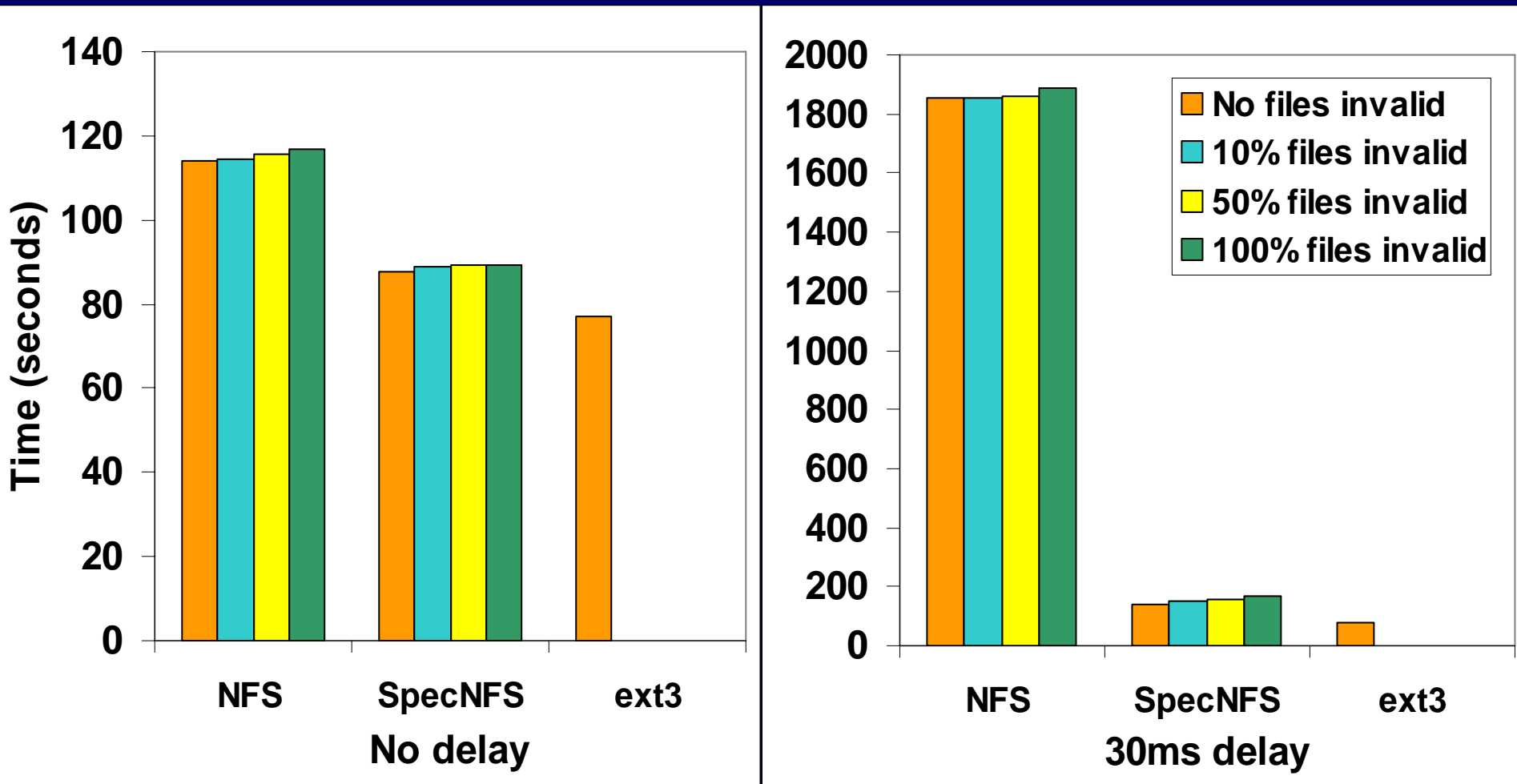
- Motivation
- Implementing speculation
- Multi-process speculation
- Using Speculator
- Evaluation

Apache Benchmark



- SpecNFS up to 14 times faster

The Cost of Rollback

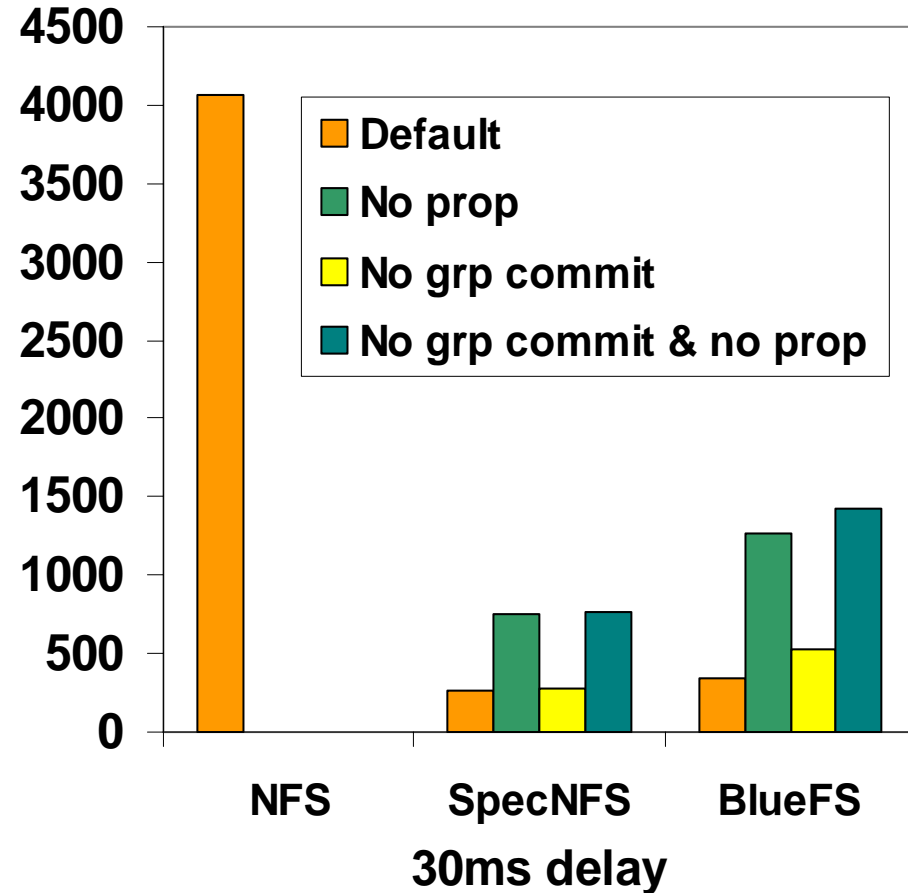
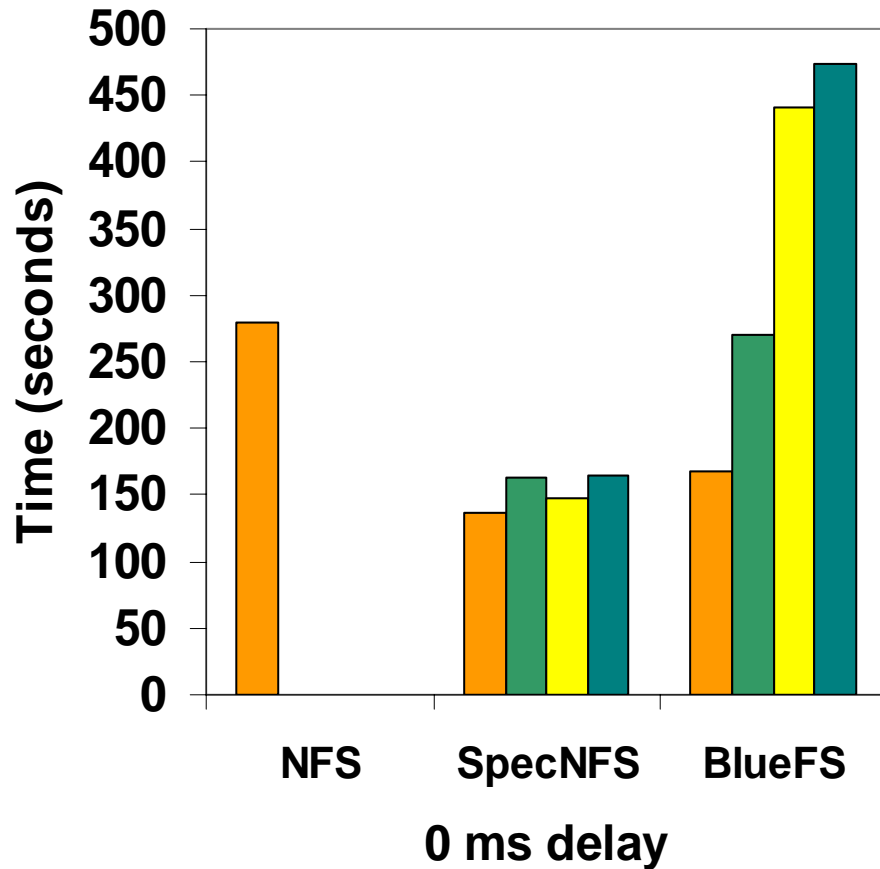


- All files out of date SpecNFS up to 11x faster

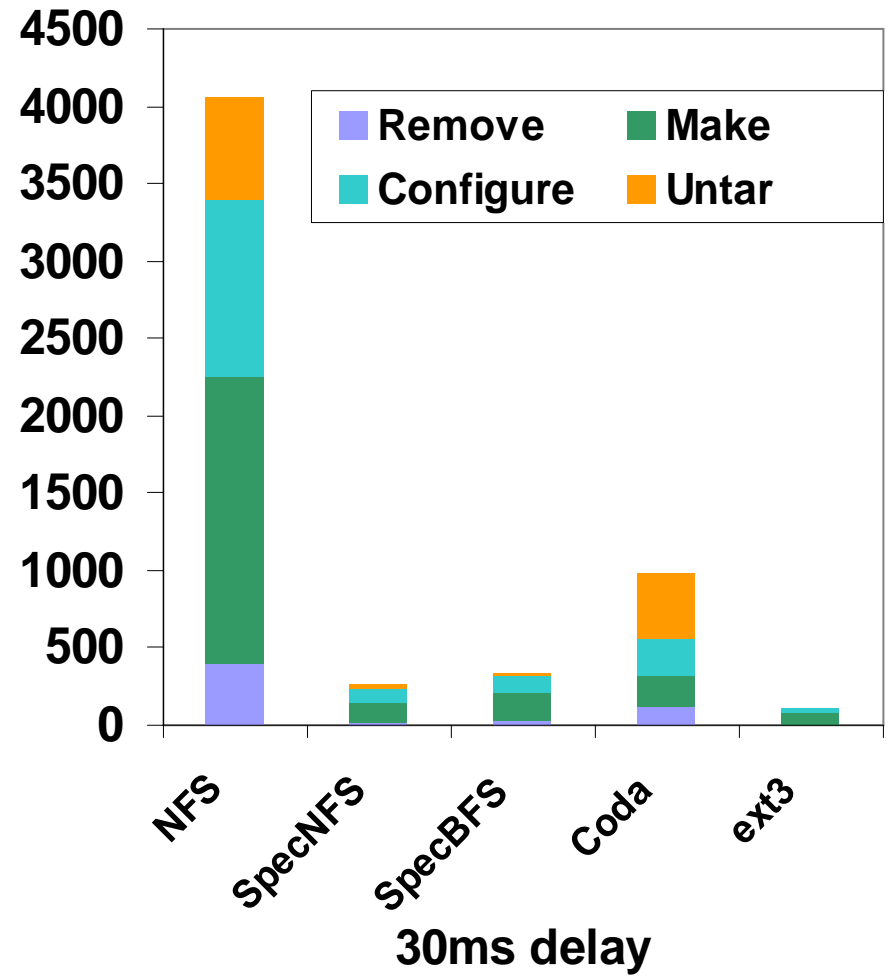
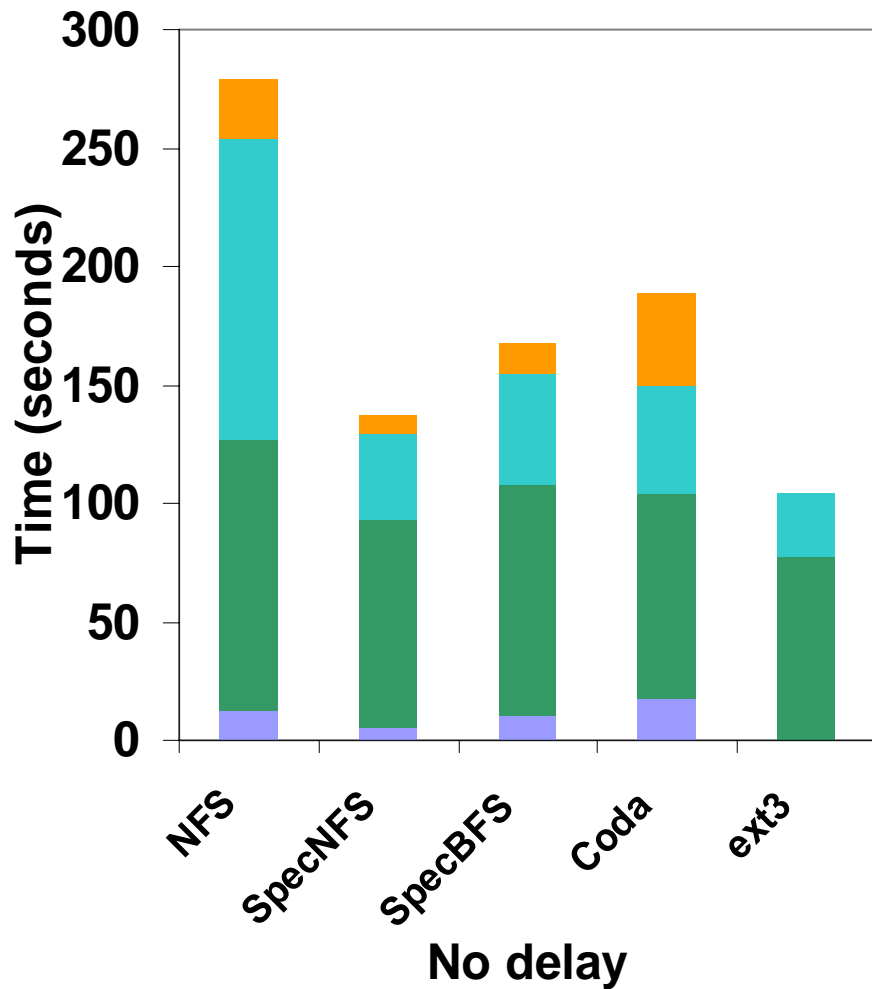
Conclusion

- Speculator greatly improves performance of existing distributed file systems
- Speculator enables new file systems to be safe, consistent and *fast*

Group Commit & Sharing State



Apache Benchmark



Related Work

- Chang & Gibson, Fraser & Chang
 - Speculative pre-fetching
- Time Warp
 - Virtual Time: distributed simulations
- Hardware branch prediction
- Transactional file systems