Speculations:
Speculative Execution in a Distributed File System[1]
and
Rethink the Sync[2]

Edmund Nightingale[12], Kaushik Veeraraghavan[2],
Peter Chen[12], Jason Flinn[12]

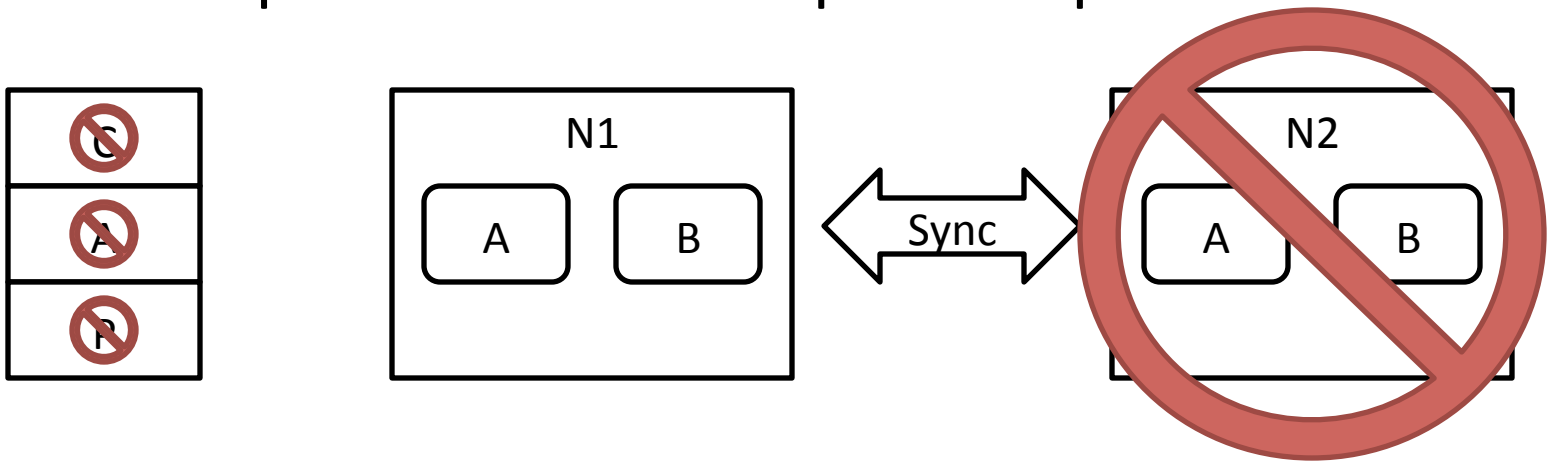
Presentation by Ji-Yong Shin
(Some slides are from Nightingale's talk)

Agenda

- CAP theorem, Consistency Semantics, Consistency Model
- Papers
  - Speculative Execution in a Distributed File System (Award Paper from SOSP'05)
  - Rethink the Sync (Best Paper from OSDI'06)

CAP Theorem by Eric Brewer

- At most two of CAP can be satisfied simultaneously
  - Consistency: correctness of data
  - Availability: guaranteed immediate access to data
  - Partition Tolerance: guaranteed functioning despite network disruption or partition



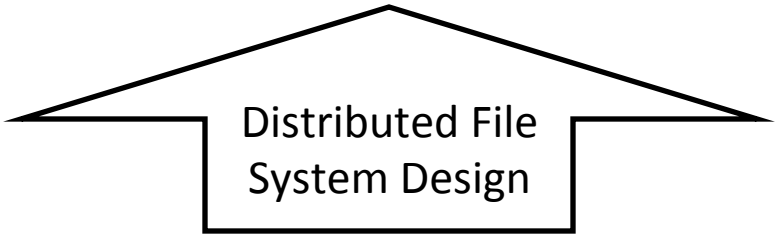
ACID vs BASE
(Not exactly opposite but..)

ACID

- Atomicity
- Consistency
- Isolation
- Duration

BASE

- Basically Available
- Soft-state
- Eventually consistent



Consistency Semantics by Leslie Lamport

- Atomic (single copy)
  - Every read returns the value of the most recent write
- Regular
  - Read not concurrent with any write returns the most recent write
  - Read concurrent with some writes returns either the most recent write or a value of concurrent write
- Safe
  - Read not concurrent with any write returns the most recent write
  - Read concurrent with some writes returns any value

Consistency Models

- Strict consistency
  - All executions in strict order
- Sequential consistency
  - All execution results exposed in strict order
- Causal consistency
  - All executions results with causal dependency exposed in strict order

- Close-to-open consistency
  - All execution results of processes that closed the file should be exposed to process opening the file

- Delta consistency
  - After fixed period of time all memory parts will be consistent
- Eventually consistency
  - After sufficiently long period of time all memory parts will be consistent

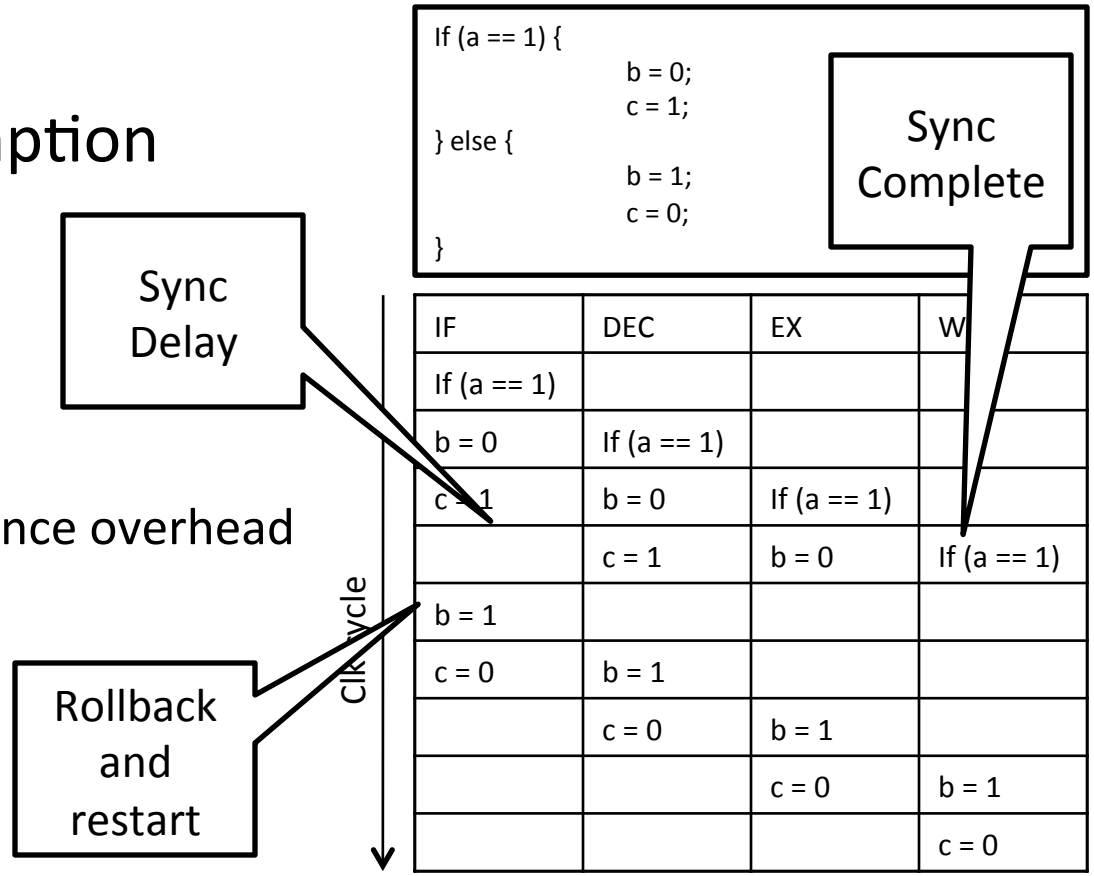
- Consistency and CAP theorem
- Papers
  - **Speculative Execution in a Distributed File System (Award Paper from SOSP'05)**
  - Rethink the Sync (Best Paper from OSDI'06)

Authors

- Edmund B Nightingale
  - PhD from UMich (Jason Flinn)
  - Microsoft Research
  - Both papers are part of PhD Thesis
- Kaushik Veeraraghavan
  - PhD Student in Umich (Jason Flinn)
- Peter M Chen
  - PhD fromUCB (David Patteron)
  - Faculty at UMich
- Jason Flinn
  - PhD at CMU (Mahadev Satyanarayanan)
  - Faculty at Umich

Speculation

- Execute using assumption
  - If assumption holds
    - performance gain
  - If assumption fails
    - Restart execution
    - Not much performance overhead
- Example
  - Branch prediction
  - Transaction
  - Thread level speculation in multiprocessor (or multicore)

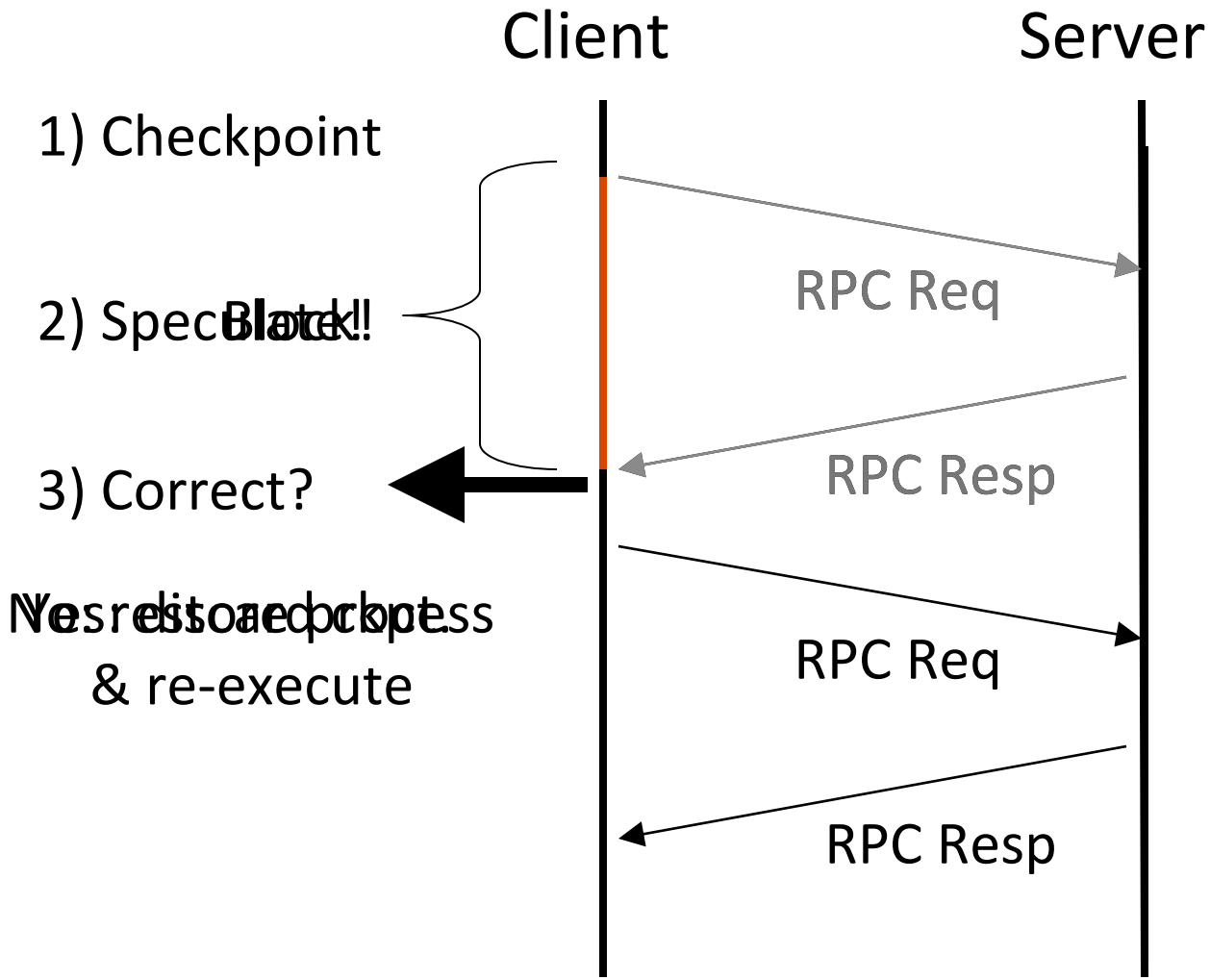


Motivation and Approach

- Distributed file system
  - Significant cost for consistency and safety
    - Block and wait from sync msg and write
  - Tradeoff between consistency and performance
    - Weak consistency for high performance

- Speculative distributed file system
  - Execute sync operations in async manner
  - While syncing execute next operation on cached files
  - Check correctness later and rollback if necessary
  - Guarantee single copy semantics

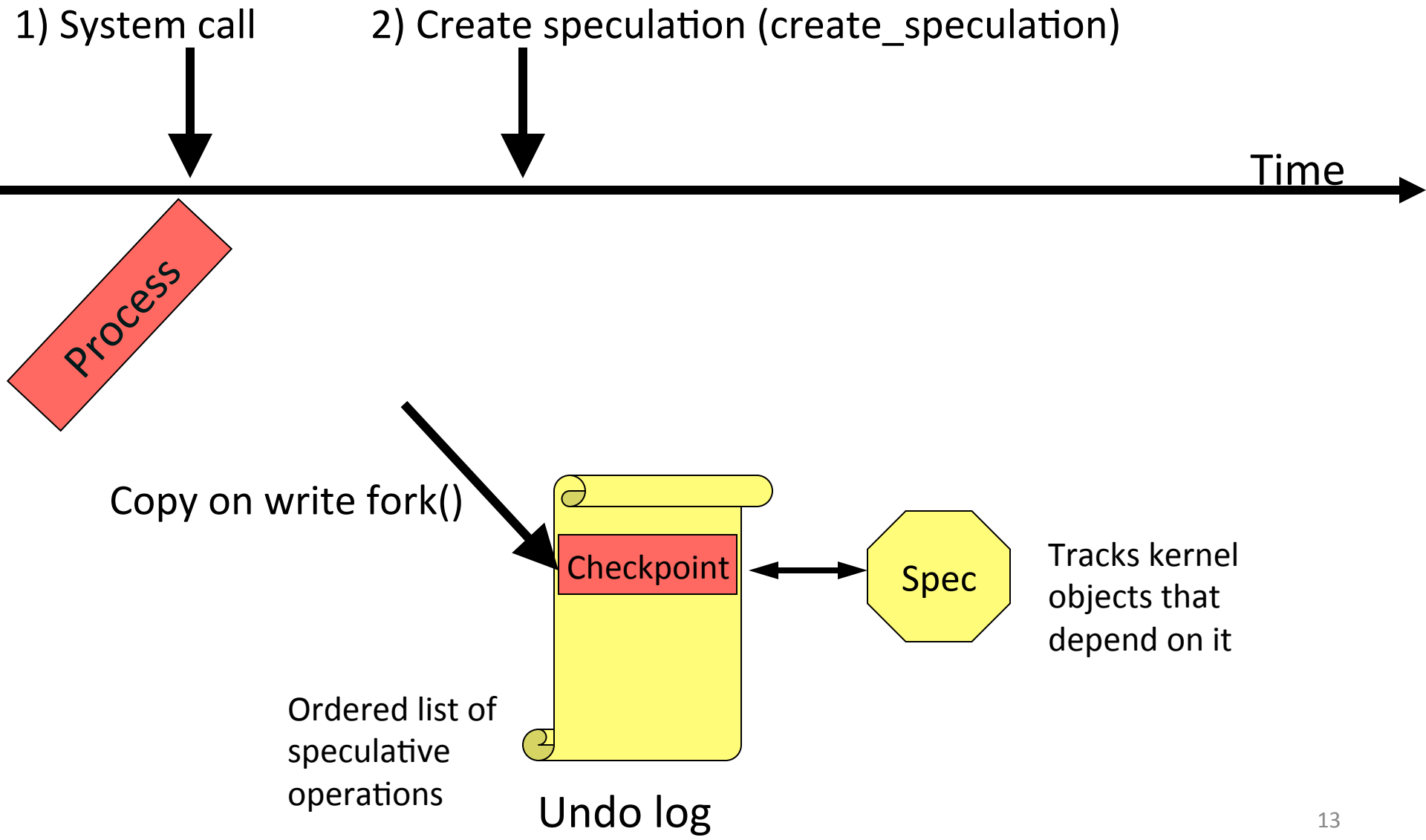
Big Idea: Speculative RPC



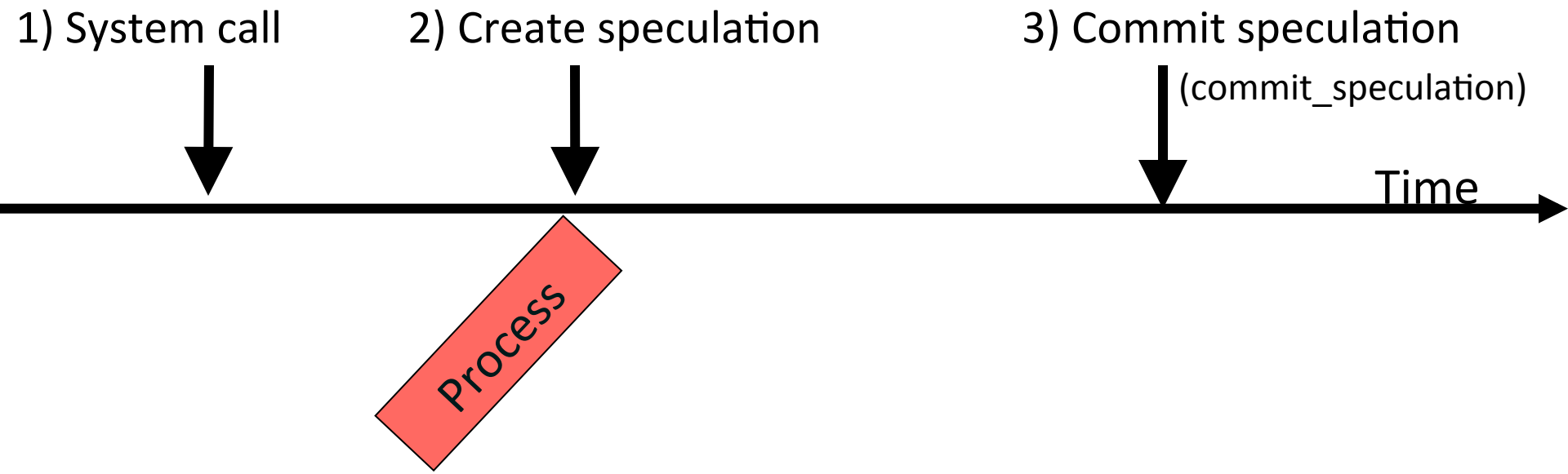
Conditions for Success

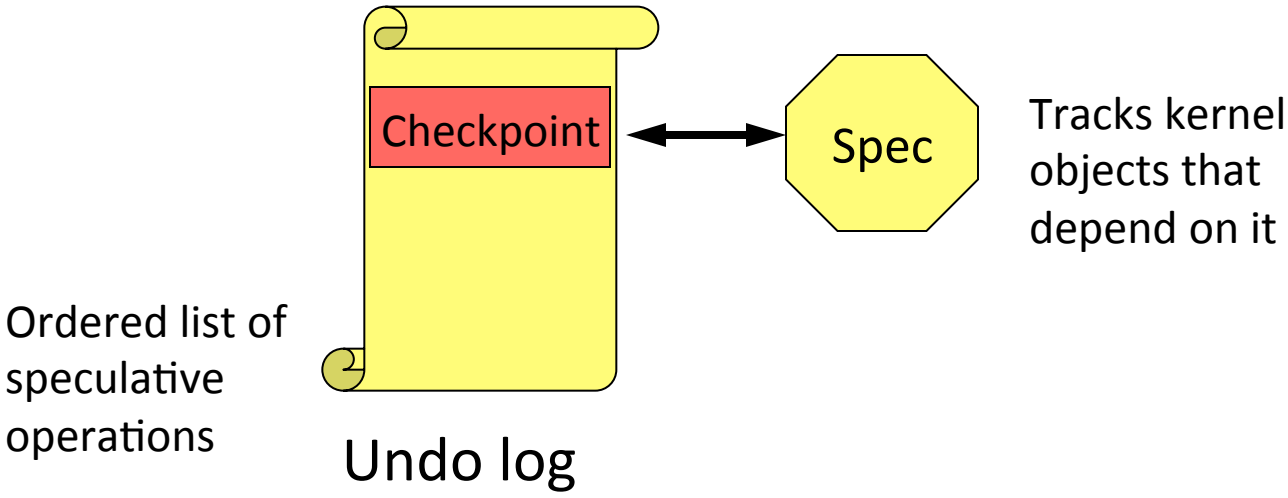
1. Highly predictive operations
   - Misprediction can worsen performance
     - Rare misprediction
2. Faster checkpointing compared to remote IO
   - Slow checkpointing is not worth doing
     - 52us for small process < network IO
3. Available spare resource for speculation
   - Speculation requires memory and CPU cycles
     - Modern computers have abundant resource

Implementing Speculation

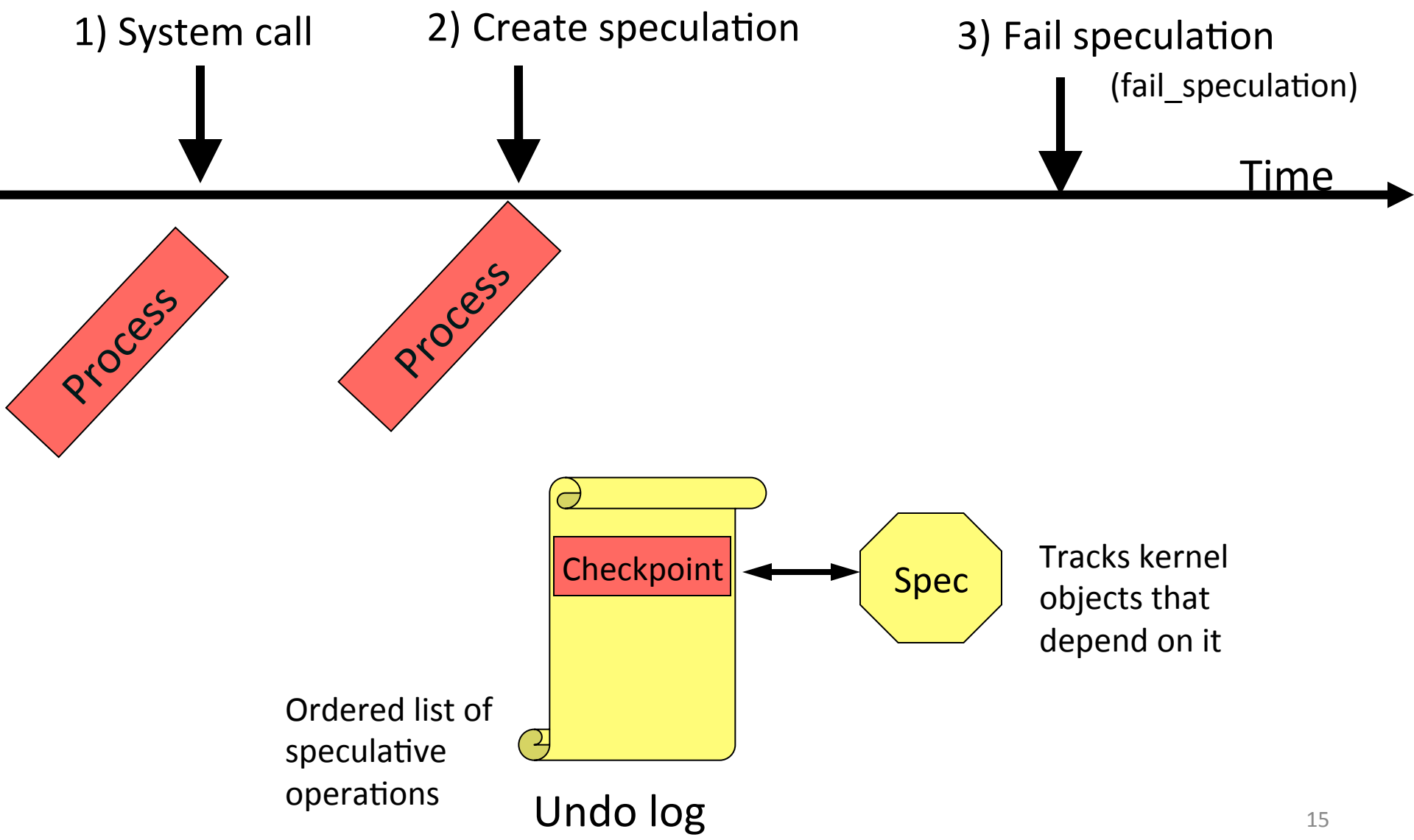


Speculation Success





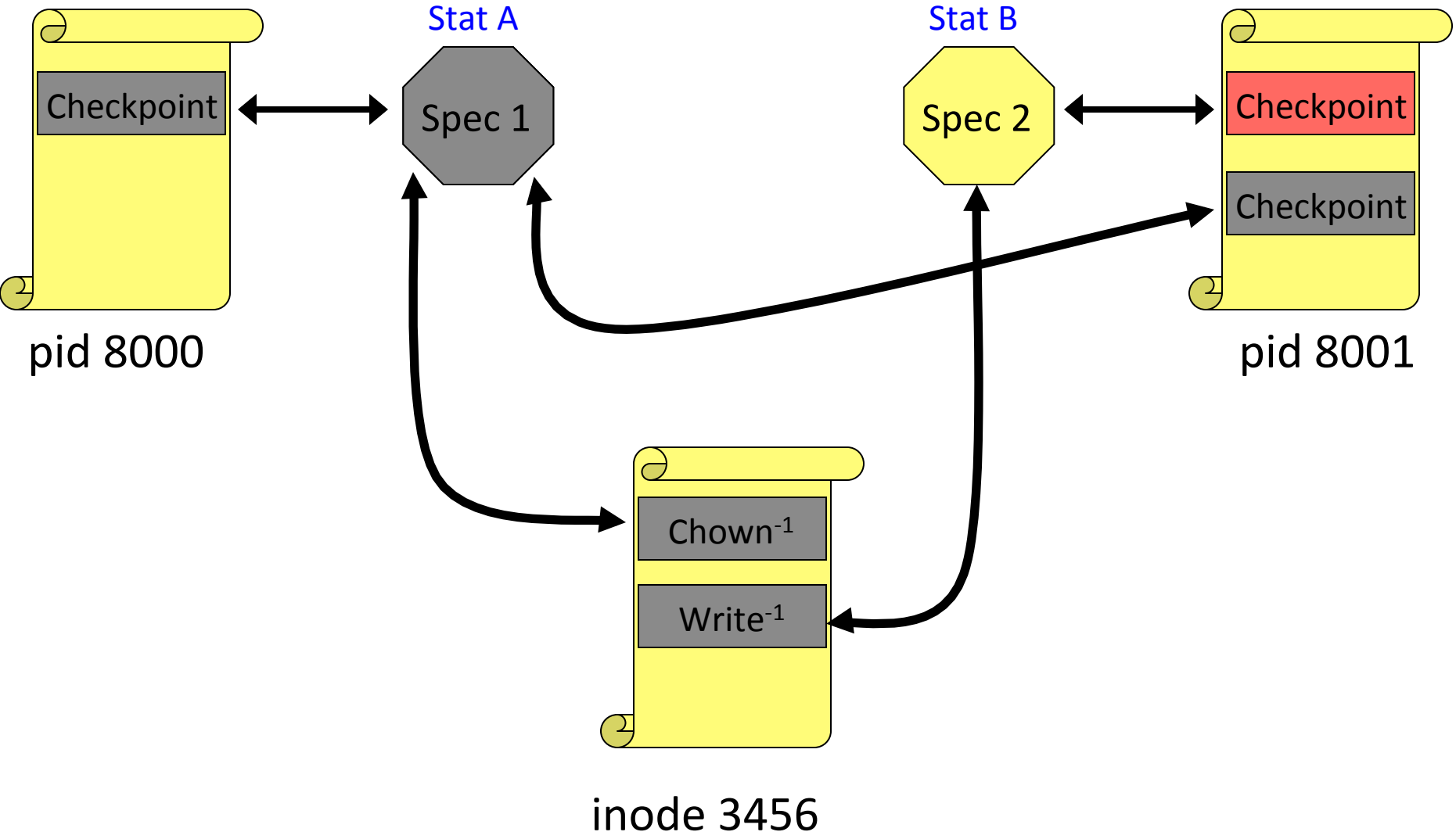
Speculation Failure



Multi-Process Speculation

- Processes often cooperate
  - Example: “make” forks children to compile, link, etc.
  - Would block if speculation limited to one task
- Supports
  - Propagate dependencies among objects
  - Objects rolled back to prior states when specs fail

Multi-Process Speculation



Ensuring Correctness

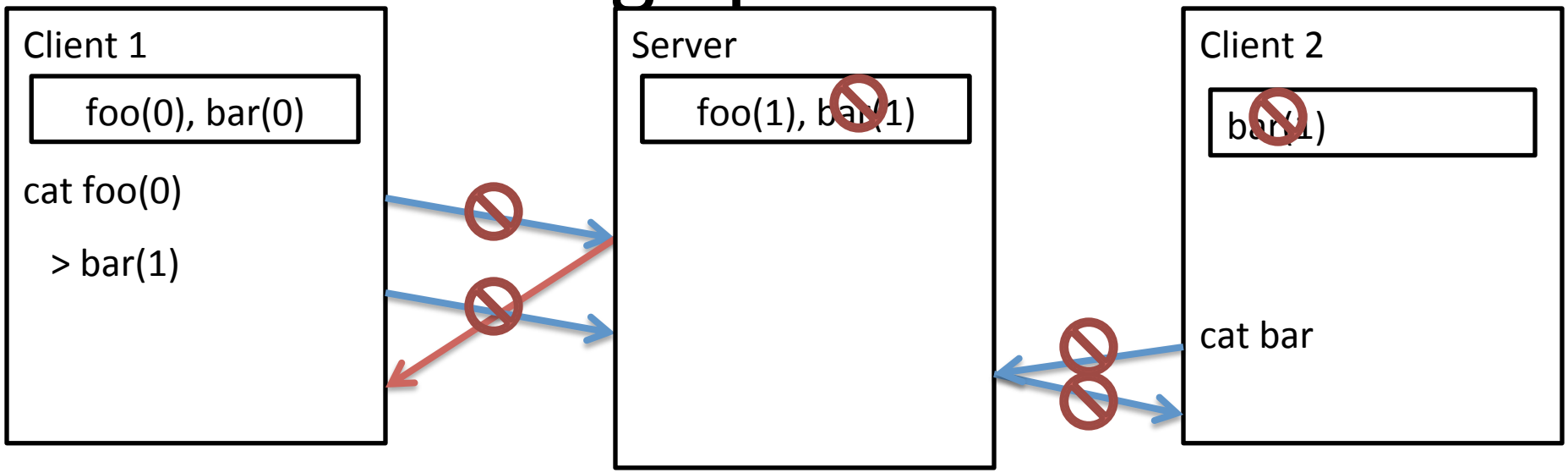
- Speculative state must **never be visible** to
  1. User or external device
  2. Process not depending on the it

- Controlling speculative process
  - Block access to external environment
    - Read only (getpid) and private state updates (dup2) allowed
  - Buffer write to external device
  - Propagate speculation if necessary

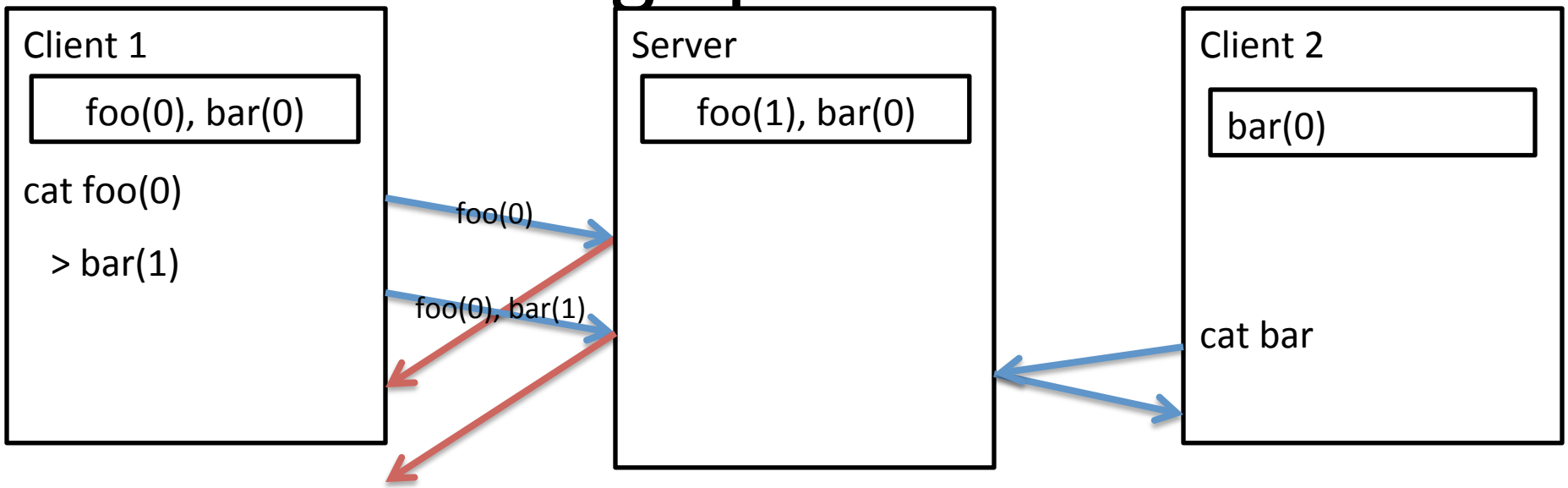
Multi-Process Speculation

- Supports
  - Objects in distributed file system
    - Will be explained in next slides
  - Objects in local memory file system (RAMFS)
  - Objects in local disk file system
    - Use buffering strategy for speculation
    - Shared on-disk metadata: only valid state committed using redo and undo
    - Journal: only commit non speculative operations
  - Etc
    - Pipe, fifos, unix sockets, signals, fork, exit
- Doesn't support
  - write-shared memory including V IPC, futex

Using Speculation



Using Speculation

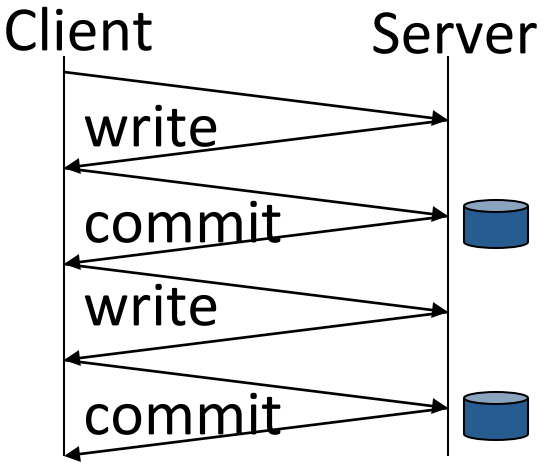


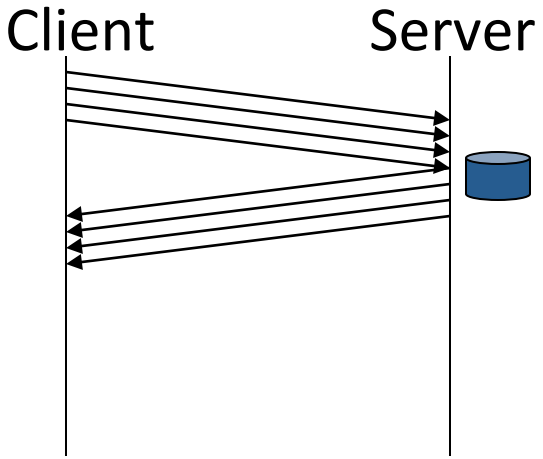
- Mutating Operation
  - Server determines speculation success/failure
    - **State at server never speculative**
    - Can be durable to server crash
  - Requires server to track failed speculations
  - Requires in-order processing of messages

Group Commit

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

Updating different files…





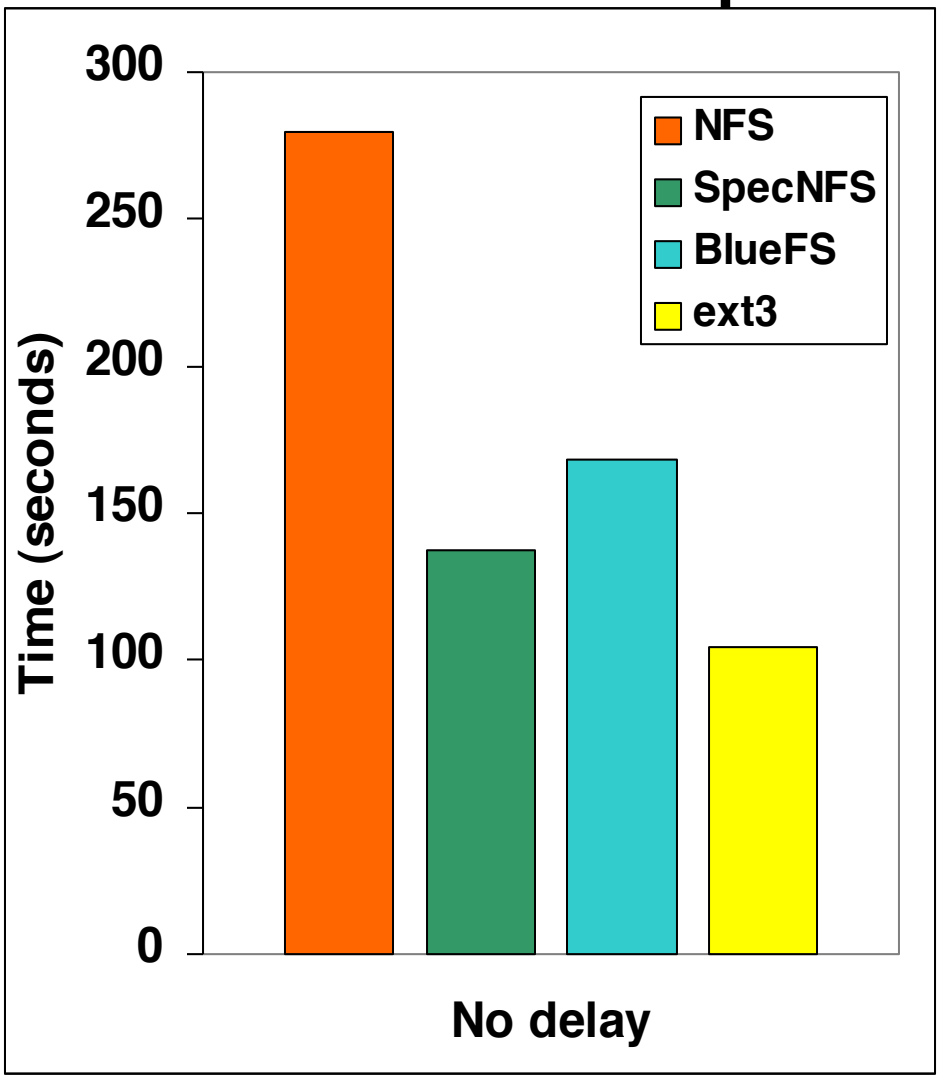
Can significantly improve disk throughput

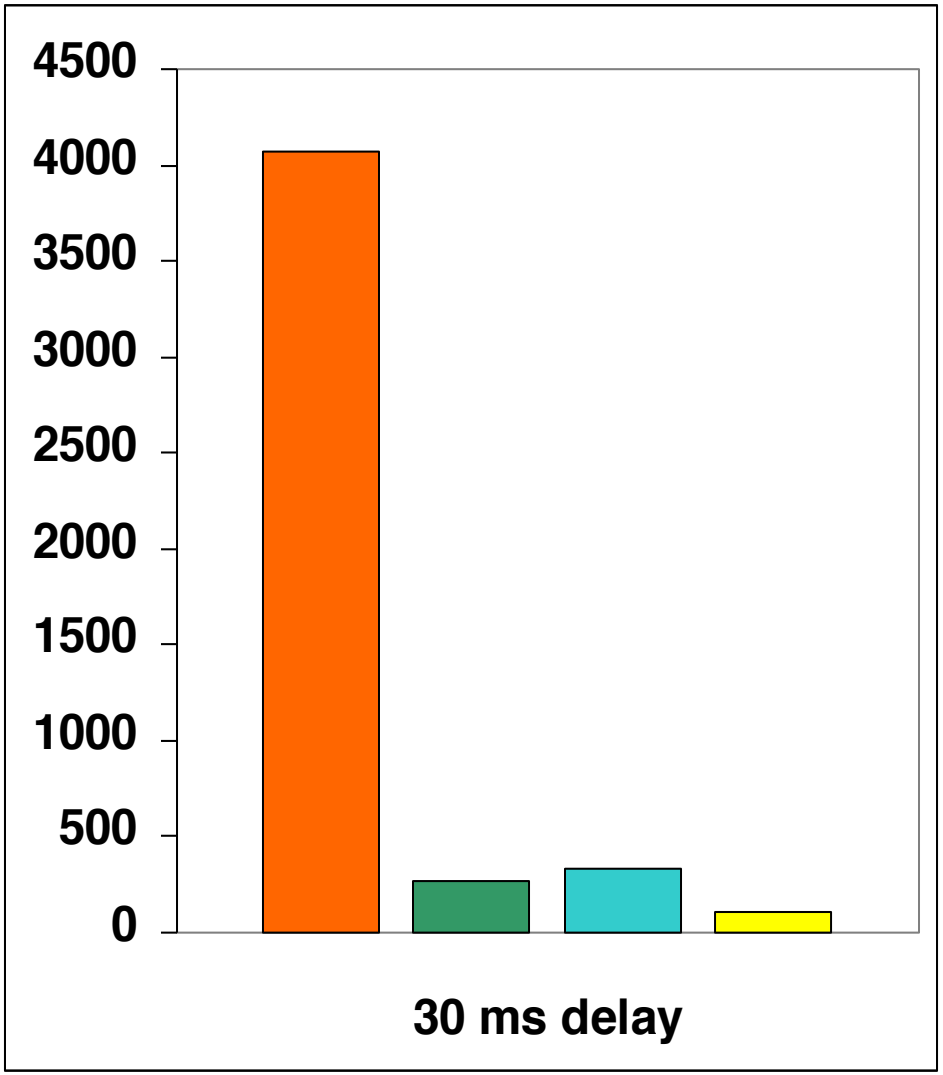
Implementation

- SpecNFS
  - Modified NFSv3 in Linux 2.4 kernel to support Speculator
    - Same RPCs issued (but many now asynchronous)
    - SpecNFS has same close-to-open consistency, safety as NFS
- BlueFS
  - new file system for Speculator
    - Single copy semantics
    - Each file, directory, etc. has version number
    - Check server for every operation

- Two Dell Precision 370 desktops as the client and file server
- Routed packet through NISTnet network emulator to insert delay.

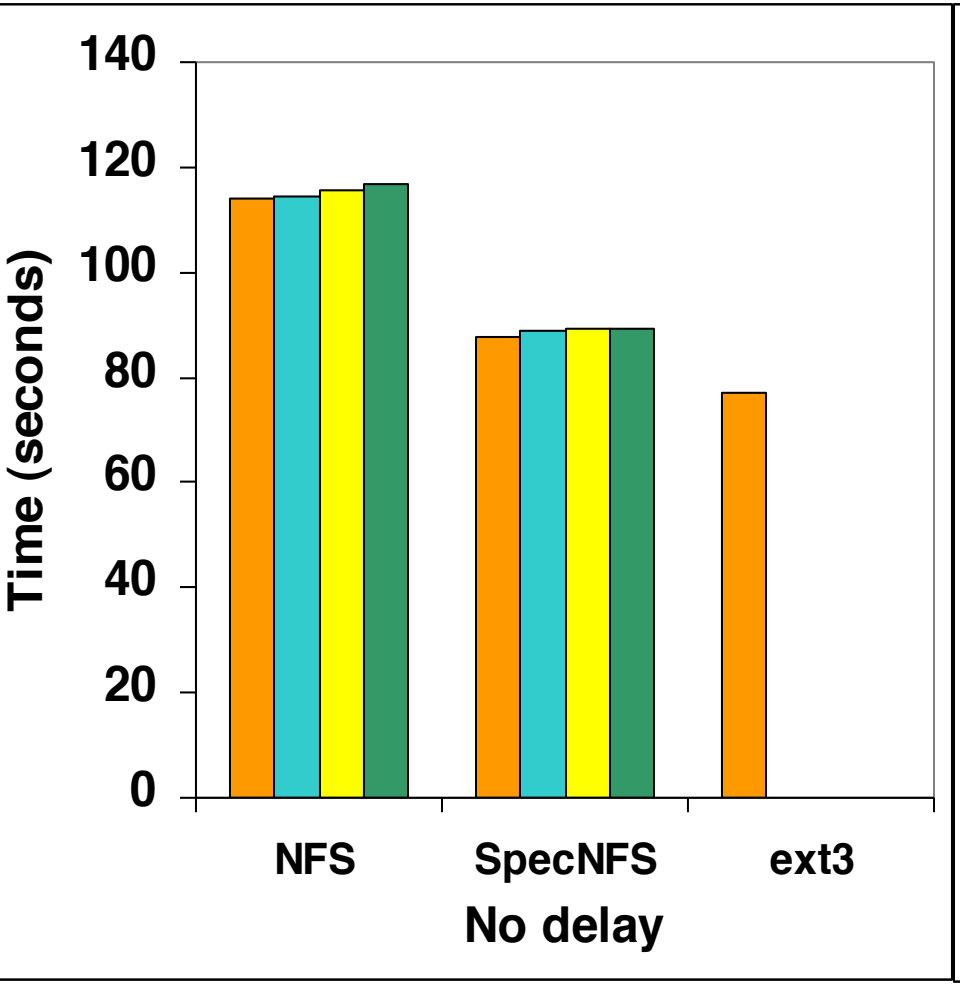
Apache Build

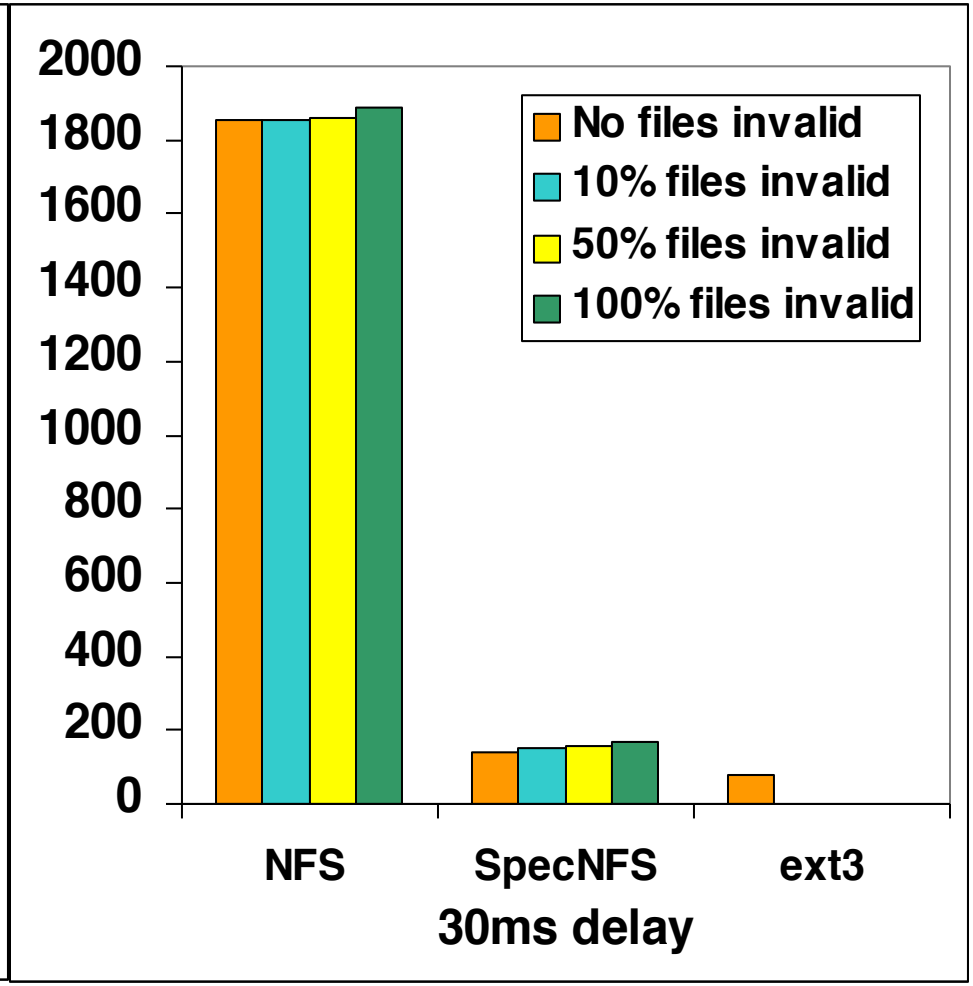




- With delays SpecNFS up to 14 times faster

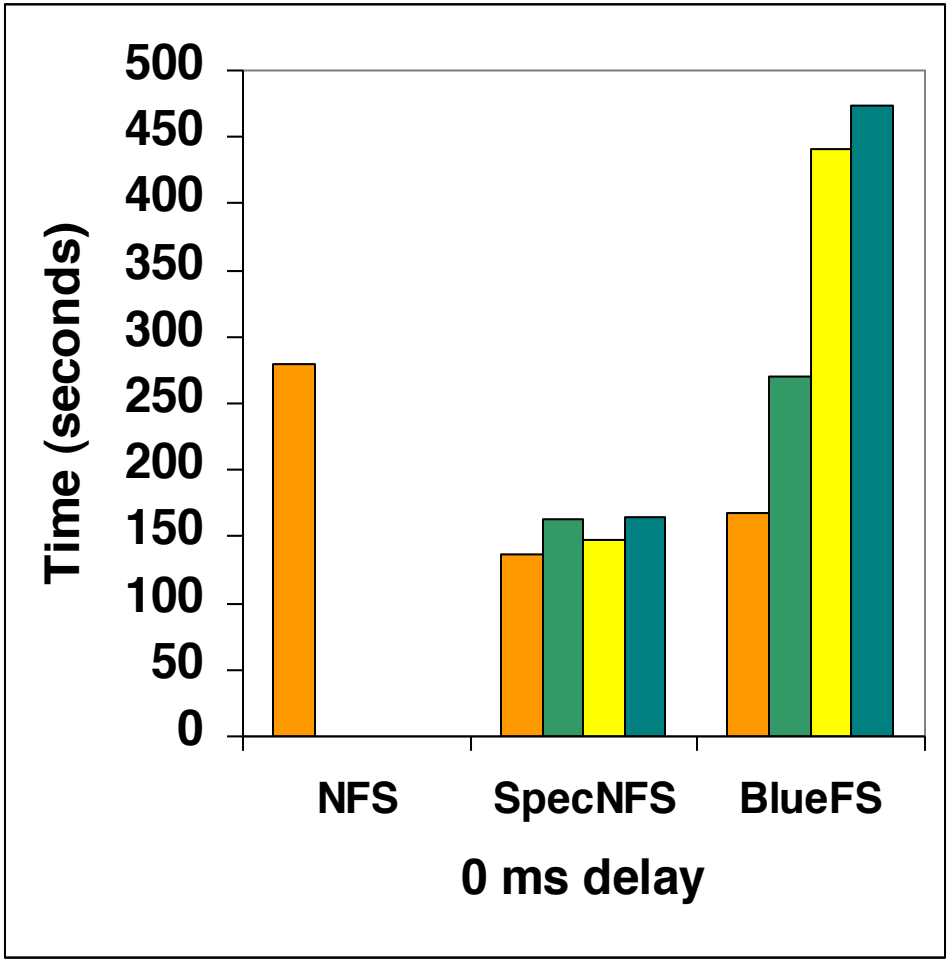
The Cost of Rollback

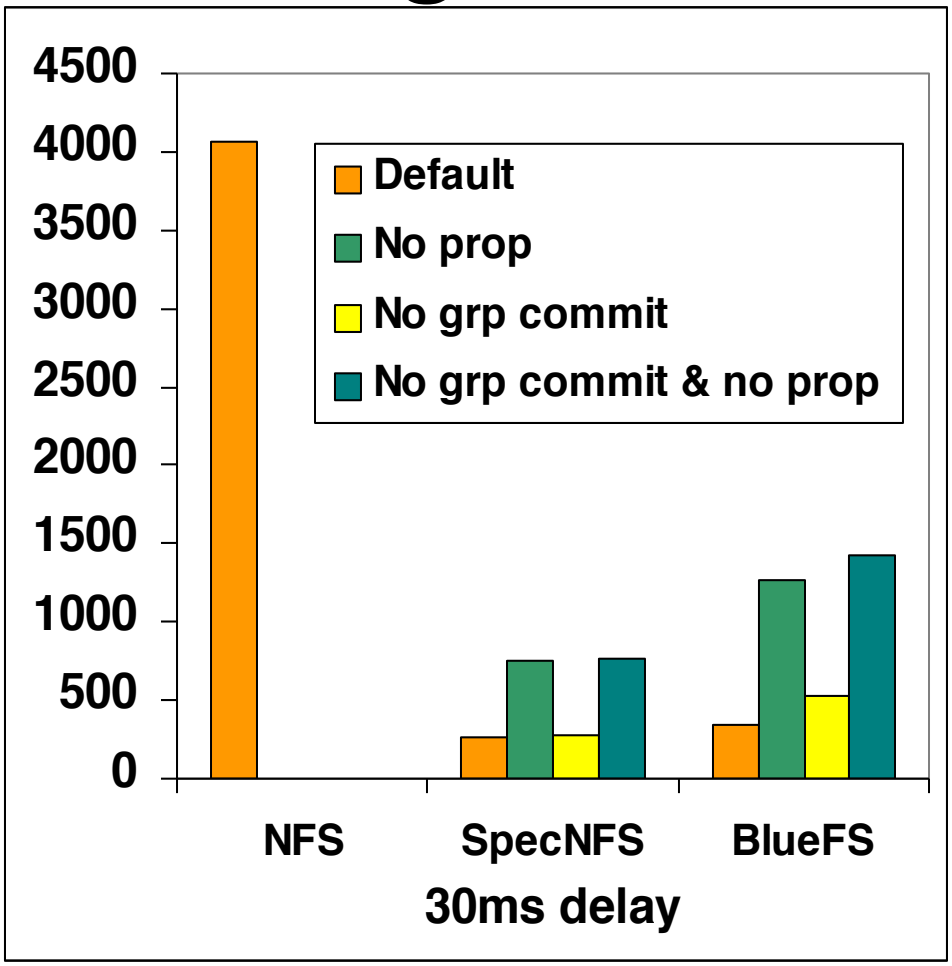




- All files out of date SpecNFS up to 11x faster

Group Commit & Sharing State





Conclusion

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

Discussion

- Starvation (Infinite rollback)?
- Overhead for maintaining speculation?
  - Memory or CPU?
- Multiple server environment?
- Consistency?
  - Consistency Semantics?
  - Consistency Model?
  - CAP Theorem?

- CAP theorem, Consistency Semantics, Consistency Model
- Papers
  - Speculative Execution in a Distributed File System (Award Paper from SOSP'05)
  - **Rethink the Sync (Best Paper from OSDI'06)**

Synchronization

Asynchronous IO

- High performance
  - Non-blocking
- Low reliability
  - Vulnerable to crash
  - Ordering not guaranteed

Synchronous IO

- Low performance
  - Blocking
- High reliability
  - Resilient to crash
  - Guaranteed ordering of IO

External Synchrony

- High performance close to async IO
  - Async-like execution until externalization
- High reliability close to synchronous
  - User centric view of guaranteed durability

External Synchrony

- Delay commit of data until externally observable operation is necessary
  - Print to screen
  - Packet send

- Externally observable behavior implicates
  - Operation before the observed behavior are committed

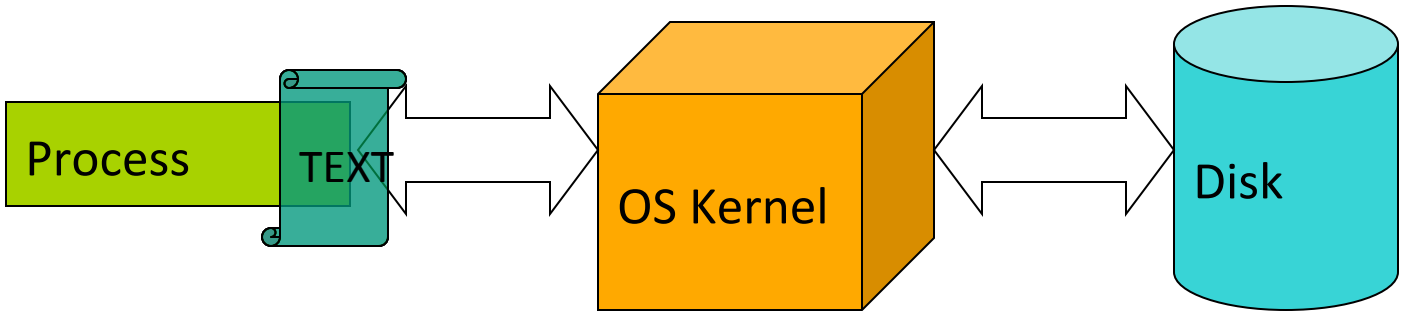
Example: Synchronous I/O

```
101   write(buf_1);
102   write(buf_2);
103   print("work done");
104   foo();
```

Application blocks

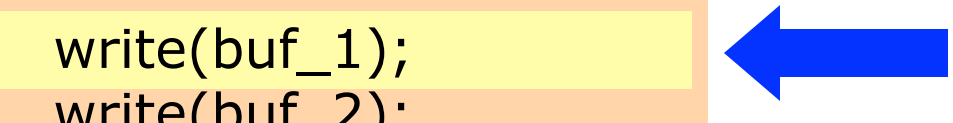
Application blocks

```
%work done
%
```

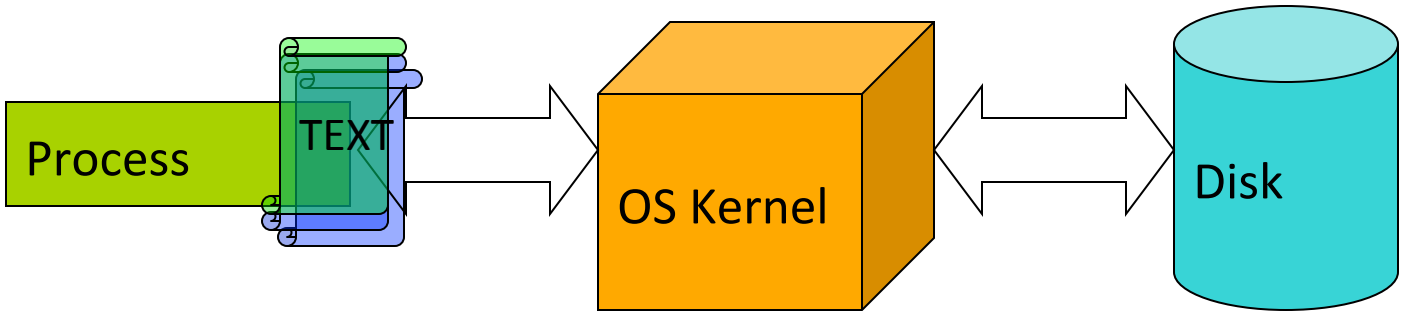


Example: External synchrony

```
101   write(buf_1);
102   write(buf_2);
103   print("work done");
104   foo();
```



```
%work done
%
```



Improving Performance

- Group commit of multiple modification
  - Atomic commit reduces disk access
- Buffering of output
  - Output function runs while committing
  - Buffered output is released after completion of commit

Multiprocess support

- Necessary functions
  - Tracking down causal dependencies
    - Speculator concept borrowed
  - Output triggered commit
    - Buffering output borrowed from Speculator

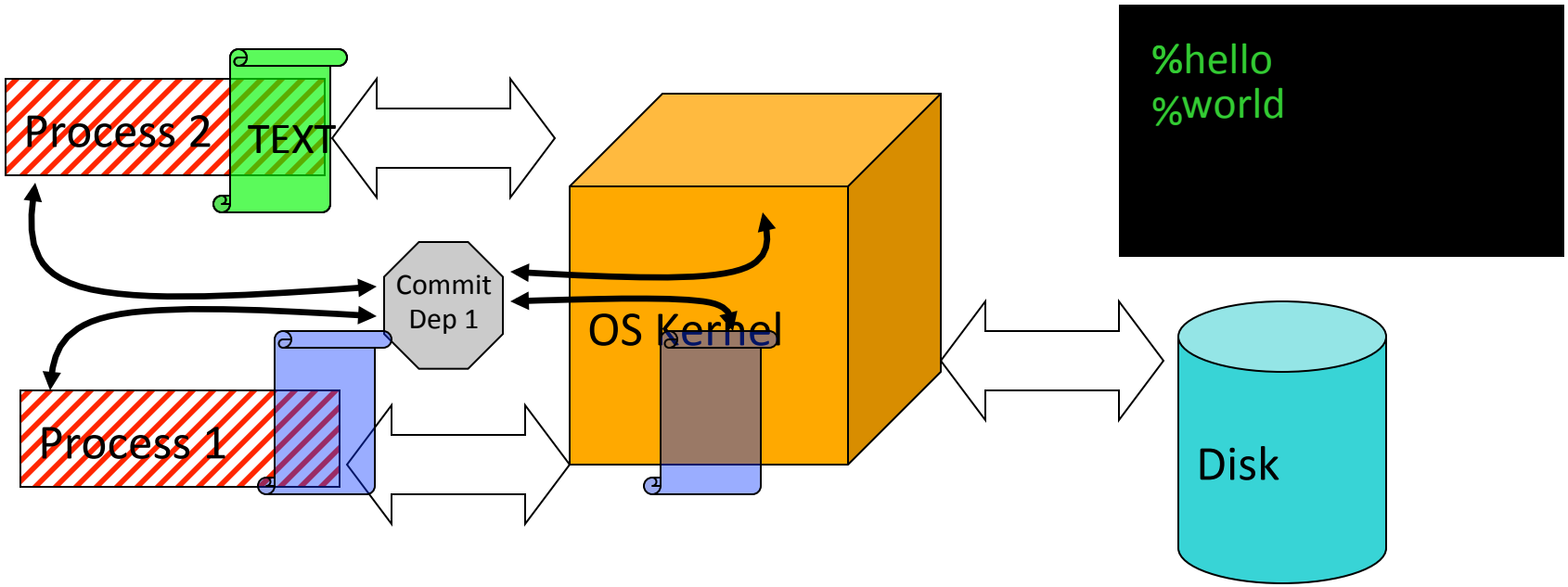
Multiprocess support

Process 1

```
101   write(file1);
102   do_something();
```

Process 2

```
101   print ("hello");
102   read(file1);
103   print("world");
```



Limitation

- Application specific recovery is difficult
  - Delayed commit makes it difficult to track back
- Commit may be unlimitedly delayed
  - 5 second rule applied, users may not meet user's expectation
- Data in multiple file system is difficult to commit in single transaction
  - Journal in different locations

Implementation

- Implemented ext sync file system Xsyncfs
  - Based on the ext3 file system and Speculator
  - Use journaling to preserve order of writes
  - Use write barriers to flush volatile cache
    - Write to disk guaranteed

Speculator

- Hide speculative state until RPC response
- Trace of causal dependency for commit and roll back
- Buffers output
- Group commit

External Synchrony

- Delay commit until externalization
- Trace of causal dependency for commit
- Buffers output
- Group commit

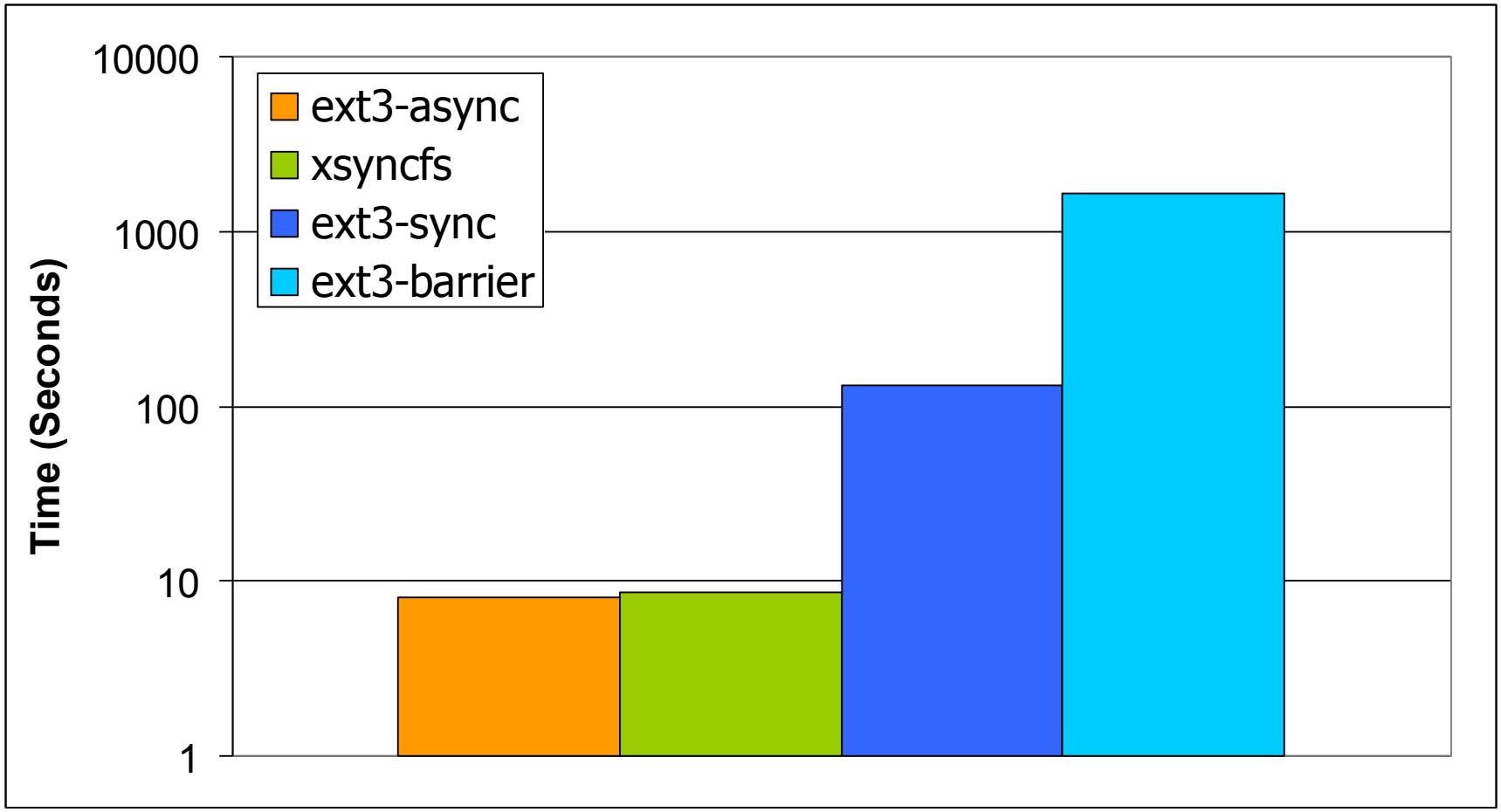
Evaluation

- Compare Xsyncfs to 3 other file systems
  - Default asynchronous ext3
  - Default synchronous ext3
  - Synchronous ext3 with write barriers

When is data safe?

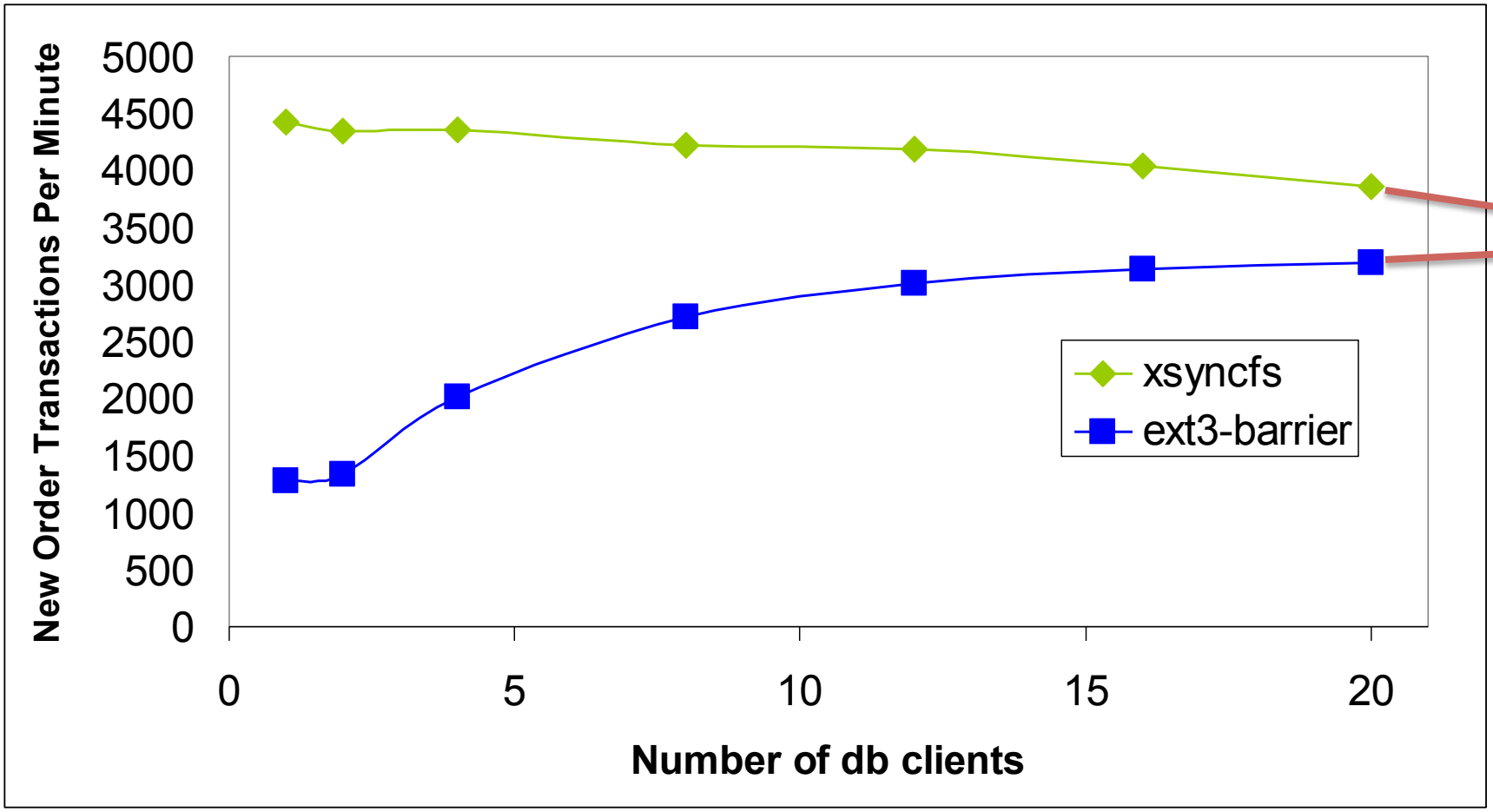
File System Configuration	Data durable on write()	Data durable on fsync()
Asynchronous	No	Not on power failure
Synchronous	Not on power failure	Not on power failure
Synchronous w/ write barriers	Yes	Yes
External synchrony	Yes	Yes

Postmark benchmark



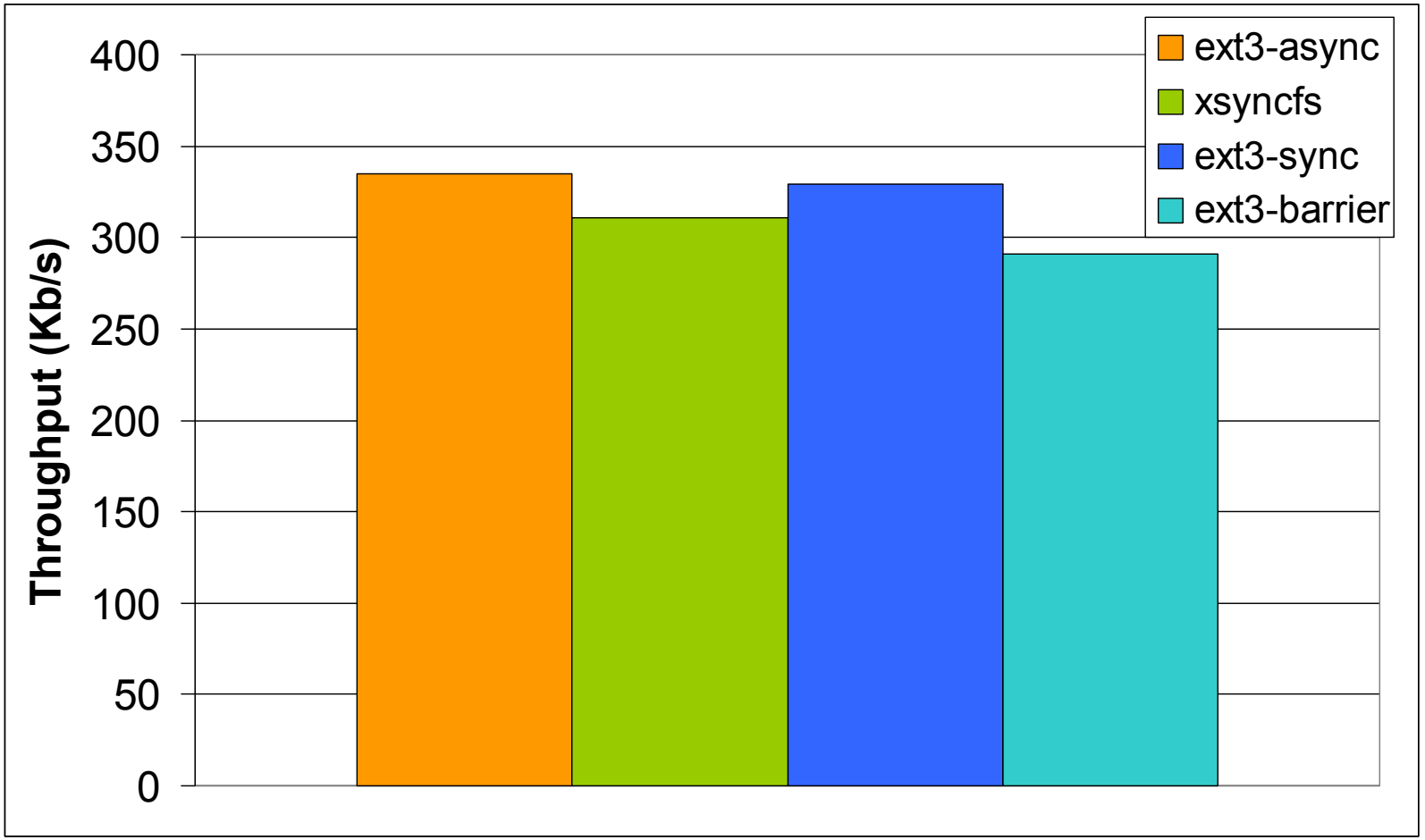
- Xsyncfs within 7% of ext3 mounted asynchronously

The MySQL benchmark



- MySQL's group commit can reach xsyncfs performance when # of client is large

Specweb99 throughput



- Xsyncfs within 8% of ext3 mounted asynchronously
- Lots of operations buffered, more externalization

Conclusion

- New concept, external synchrony, proposed
- External synchrony performs with 8% of async

Discussion

- What happens when external synchrony system fails?

- Consistency?
  - Consistency Semantics?
  - Consistency Model?
  - CAP Theorem?