6.826: Systems + Verification.

When is a system correct?

Big ideas: locking, logging, consistency, security, ...

Principled view:
- Properties
- Specifications
- Proof

Motivation: bugs.

Complex.
- Concurrency
- Distributed
- Security
- Faults
- Performance
- Evolution

⇒

Bugs
- memory safety
- deadlock
- races
- ⋮
- subtle logic

⇒

Impact
- crash
- lose data
- leak data
- wrong result.

Example: fault tolerance

Power failure ←
Disk died ←
Bad memory ←

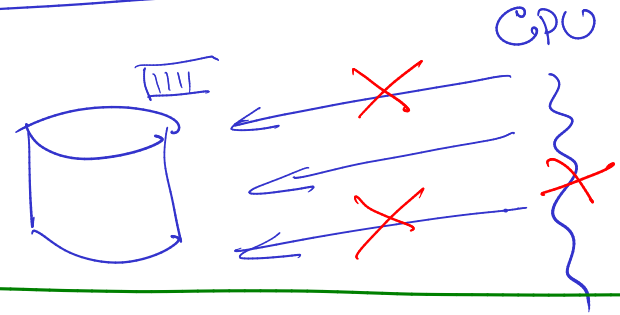
Failure models

Fail-stop
"Byzantine"

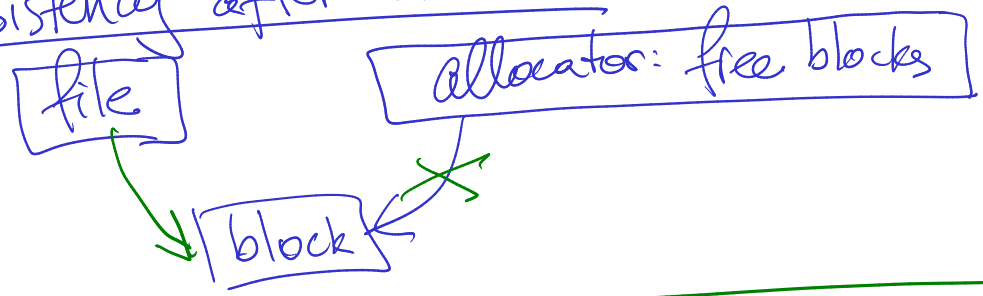
Contexts

FS } one server
DB }
Distributed DB, KV
SSD FTL

Key challenge: crash-safety persistent state



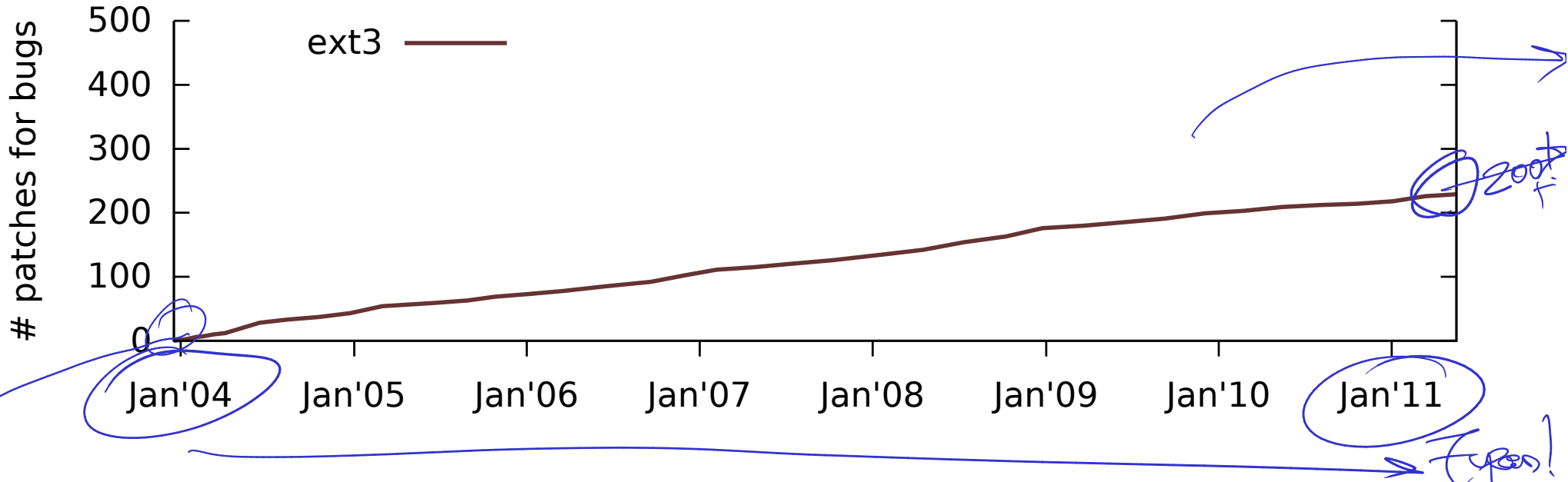
Consistency after crash?



Concurrency
Complex optimizations.
Partial failures.

Example: file systems

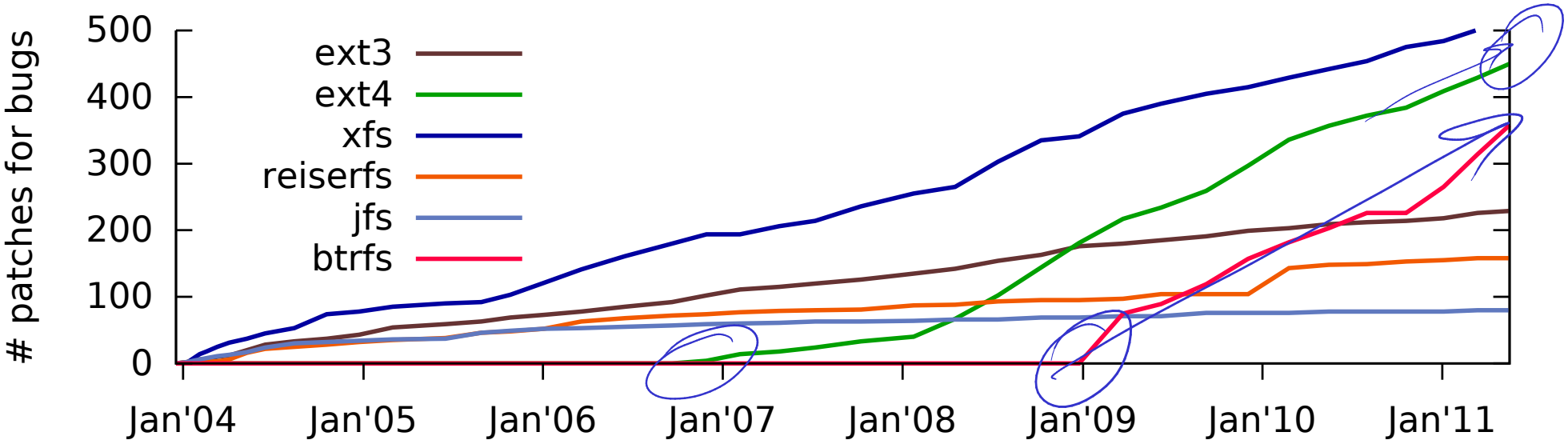
File systems are complex (e.g., Linux ext4 is ∼60,000 lines of code) and have many bugs:



Cumulative number of patches for file-system bugs in Linux; data from [Lu et al., FAST'13]

Example: file systems

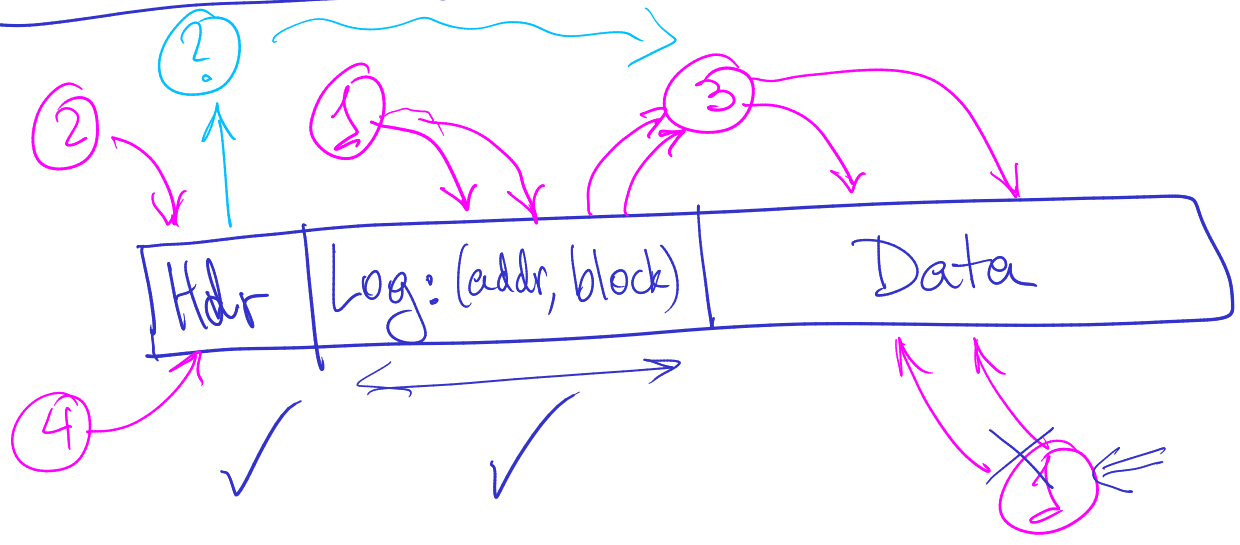
File systems are complex (e.g., Linux ext4 is ∼60,000 lines of code) and have many bugs:



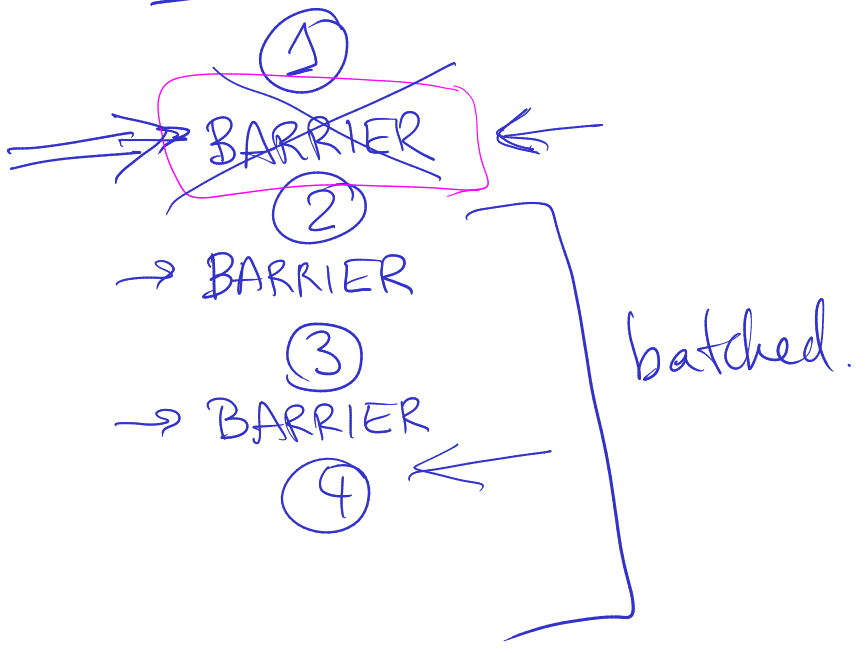
Cumulative number of patches for file-system bugs in Linux; data from [Lu et al., FAST'13]

New file systems (and bugs) are introduced over time

Write-ahead logging



Order.

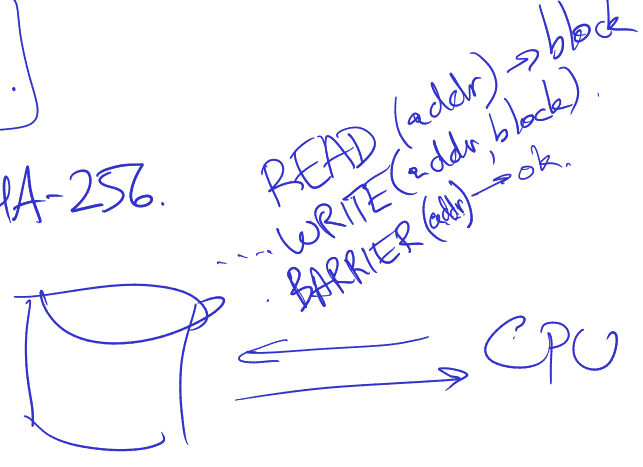


Optimization 1: disk buffering. BARRIER.

Optimization 2: log-bypass writes. ⇐

Optimization 3: checksum logging. Hdr = checksum (log).

CRC 32 SHA-256.



Bug example: data corruption

```
commit 353b67d8ced4dc53281c88150ad295e24bc4b4c5
--- a/fs/jbd/checkpoint.c
+++ b/fs/jbd/checkpoint.c
@@ -504,7 +503,25 @@ int cleanup_journal_tail(journal_t *journal)
                spin_unlock(&journal->j_state_lock);
                return 1;
        }
+       spin_unlock(&journal->j_state_lock);
+
+       /*
+        * We need to make sure that any blocks that were recently written out
+        * --- perhaps by log_do_checkpoint() --- are flushed out before we
+        * drop the transactions from the journal. It's unlikely this will be
+        * necessary, especially with an appropriately sized journal, but we
+        * need this to guarantee correctness.  Fortunately
+        * cleanup_journal_tail() doesn't get called all that often.
+        */
+       if (journal->j_flags & JFS_BARRIER)
+               blkdev_issue_flush(journal->j_fs_dev, GFP_KERNEL, NULL);

+       spin_lock(&journal->j_state_lock);
+       if (!tid_gt(first_tid, journal->j_tail_sequence)) {
+               spin_unlock(&journal->j_state_lock);
+               /* Someone else cleaned up journal so return 0 */
+               return 0;
+       }
        /* OK, update the superblock to recover the freed space.
         * Physical blocks come first: have we wrapped beyond the end of
         * the log?  */
```

Bug example: data disclosure

- Two optimizations in Linux ext4: direct data write and log checksum
- Subtle interaction: new file can contain other users' data after crash
- Bug introduced in 2008, fixed in 2014 (six years later!)

```
Author: Jan Kara <jack@suse.cz>
Date:   Tue Nov 25 20:19:17 2014 -0500

    ext4: forbid journal_async_commit in data=ordered mode

    Option journal_async_commit breaks gurantees of data=ordered mode as it
    sends only a single cache flush after writing a transaction commit
    block. Thus even though the transaction including the commit block is
    fully stored on persistent storage, file data may still linger in drives
    caches and will be lost on power failure. Since all checksums match on
    journal recovery, we replay the transaction thus possibly exposing stale
    user data.

    [...]
```

Avoiding bugs?

Testing.
- Effective
- Cannot ensure ∅ bugs.

Model checking.
- Exploring all possible states.
- State explosion

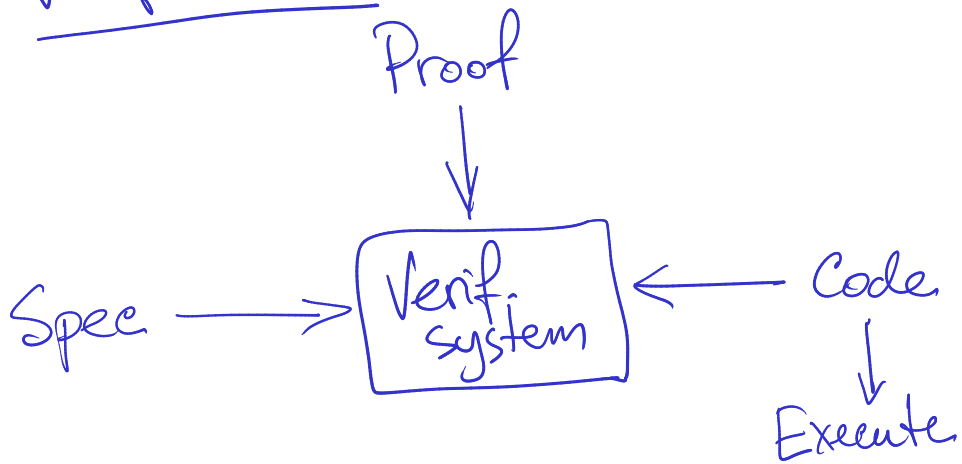
Verification
- Prove ∅ bugs.

Active area.

- Tools: Coq, Lean, Z3, CVC
- Ideas: Refinement, sep. logic, ..
- Demand: security, complex.

Sucesses: Amazon, CompCert, Chrome/Firefox crypto.

Verification



Variations.

No spec. → lightweight implied spec.
No proof → verifier automates proof.
No exec. code
No verifier.

Logistics

Papers. → Coq.

Labs. → Collab discussion.
→ Individual solution.

Grades:
- Labs
- Summaries.
- Class part.

Staff.

Butler Lampson.

Nickolai

Tej

Steller.