

Correctness of verified sys.?

Verif. \Rightarrow no bugs?

Empirical study: dist. sys.

IronFleet, Verdi, Chapar

Bugs in shims, specs, tools
Violate guarantees, crash

How to find bugs?

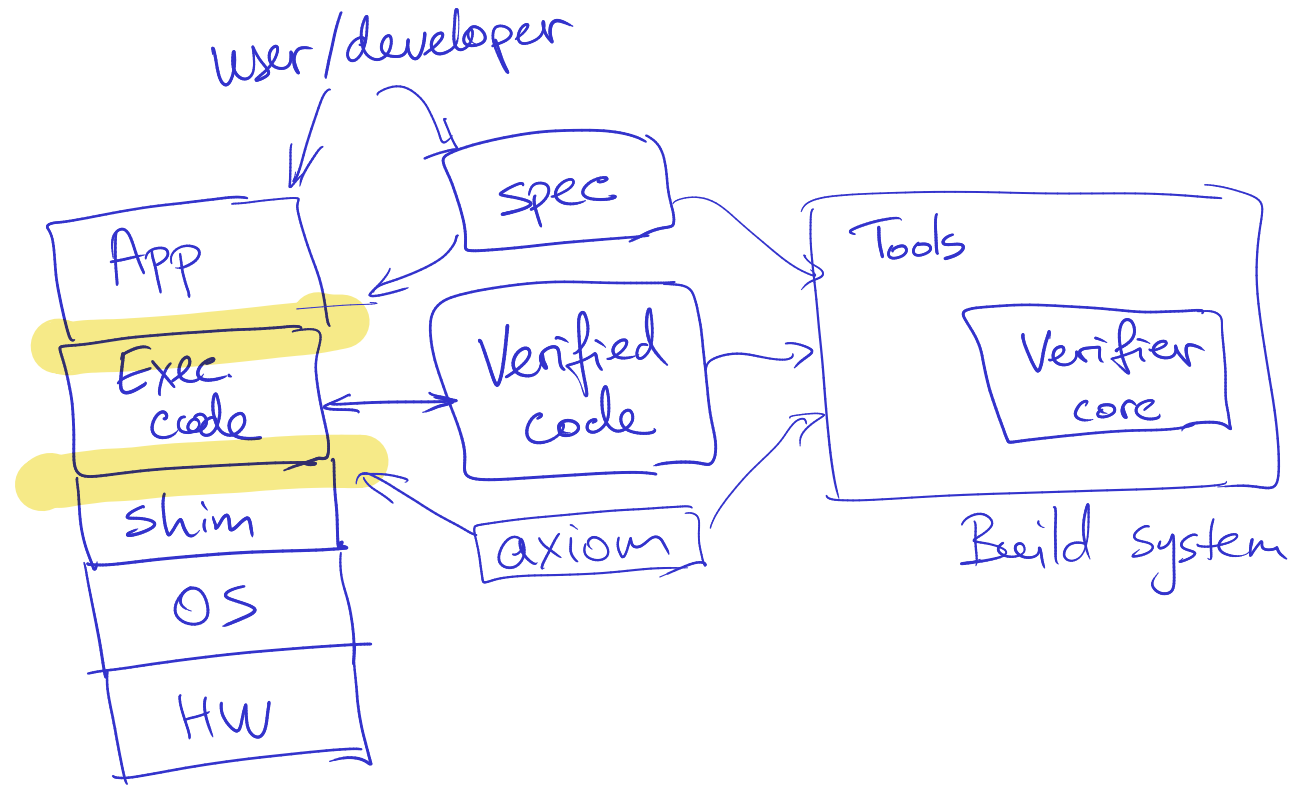
Study code, design, ...

Fuzzing shim

Cross-checking

Serious effort!

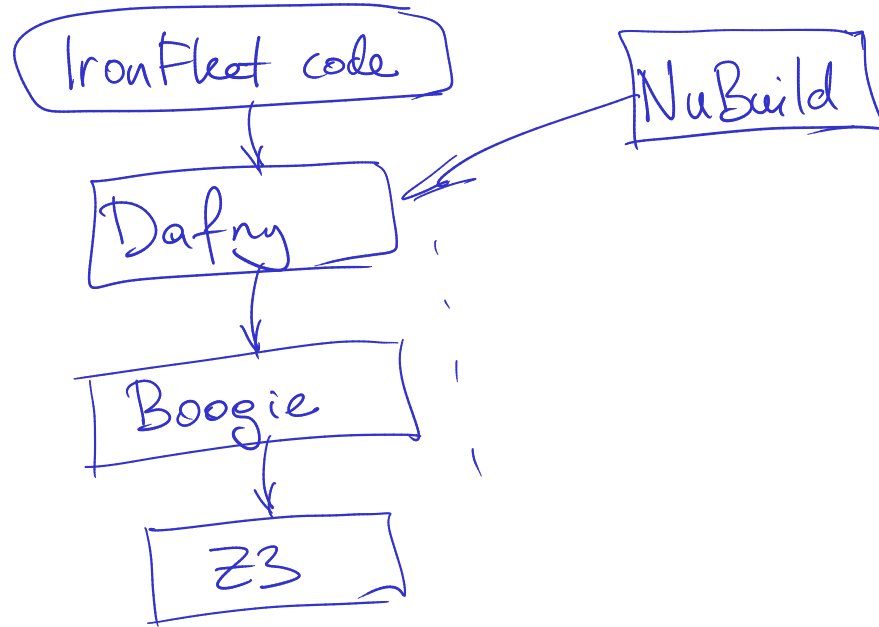
Verified sys.



Iron Fleet

RSM
Counter

Shim X
Spec ✓
Tooling ✓



Tooling bugs (I2, I3, I4)

Errors ignored.

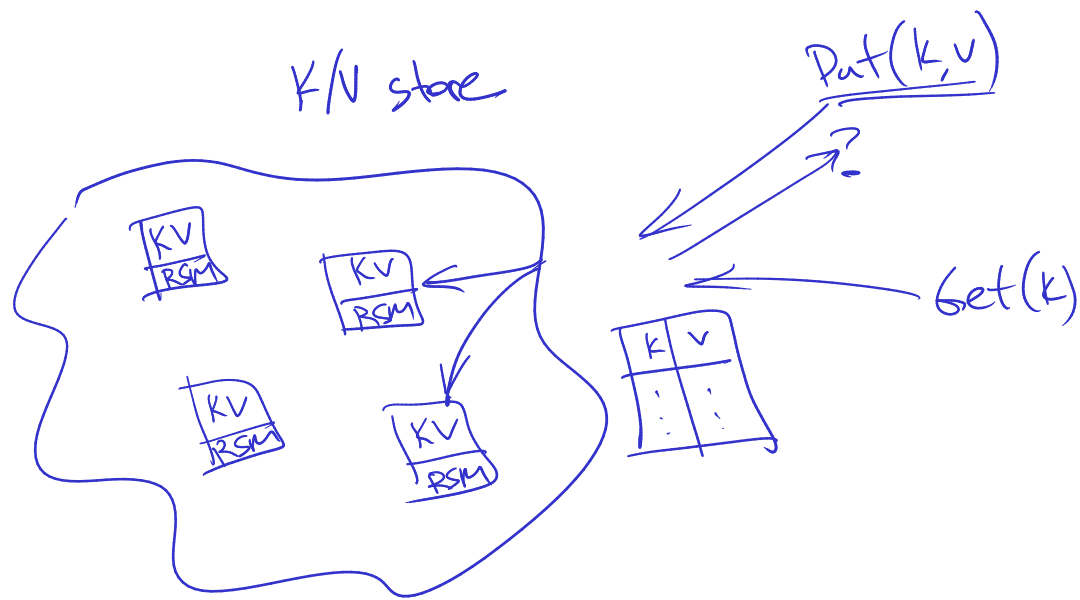
→ NuBuild parses output,
check for errors.
→ only checks first msg.

Dafny: ignores Z3 signals.

How to avoid?

Positive checks instead of errors.
Test tooling for proof failures.
Make proof checks more integral.

IronFleet spec ambiguity



Challenge: replica fails to reply?
retransmit \rightarrow duplicate exec?

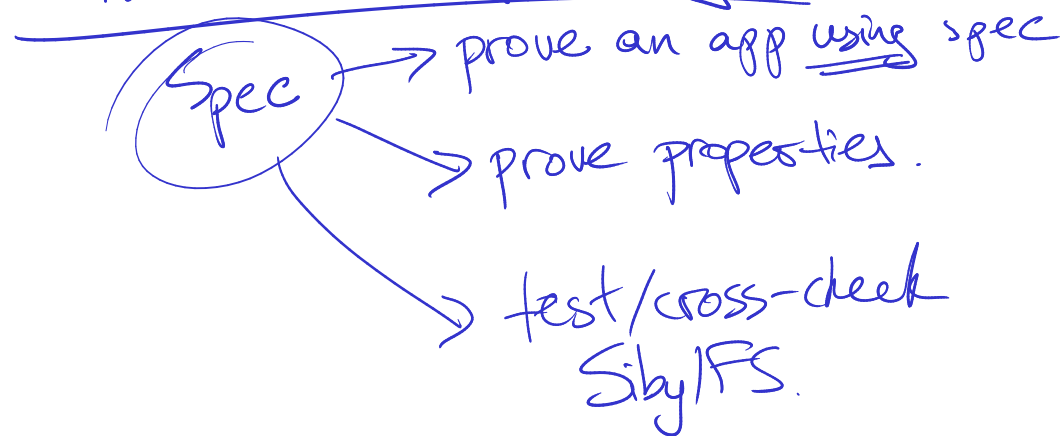
Goal: exactly-once execution \rightarrow dedup

IronFleet: code for filtering dups. \leftarrow
spec did not promise dedup.

Other spec bugs

NetCore: no output packets.
 \rightarrow FSCQ: spec vs error codes
vs Sibyl/FS.

How to avoid spec bugs?



Non-determinism

Simple.

Performance.

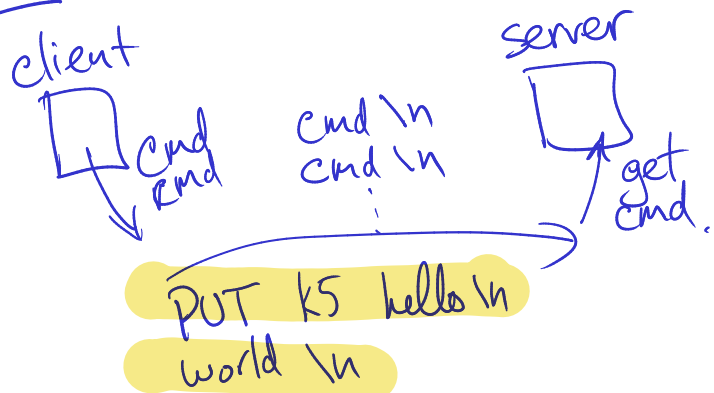
Verdi: RSM

Cog $\xrightarrow{\text{Extract}}$ Ocaml
+ shims
TCP,
FS.

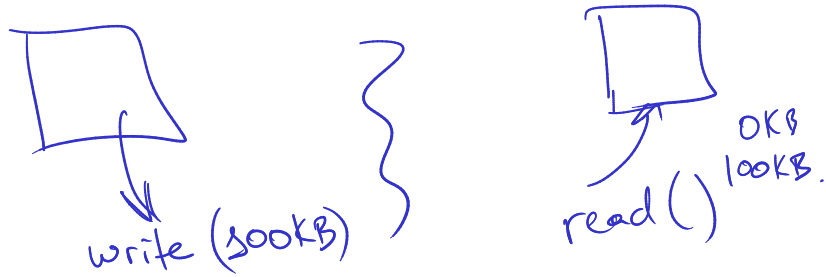
V1: recv() complete msg.



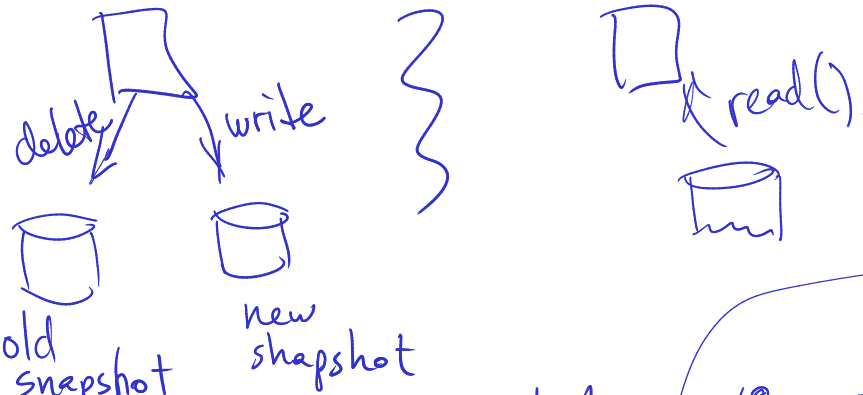
V2: incorrect marshaling



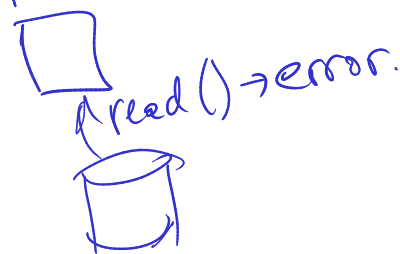
V3: large writes not atomic



V4: incorrect snapshot write order



V5: failed to read snapshot



V8: out of stack space
find Gt Index.
Cog -> Ocaml
recursive, inefficient

Chapar

KV store

Causal consistent.

UDP-based

Shim bugs!

C1: assumed no dups.

C2: assumed no drops.

C3: odd Ocarl net library.

Avoid shim bugs?

Expand verif. boundary.

Narrow waist / API → simpler spec.

Spec includes errors,
resource use, ...

Shim: clean API
well-designed libraries.

Test shim (Sibyl/FS)

Test corner cases: out of resources.

Lessons learned?

Verified code → no bugs.

Relies on assumptions

Spec → use it

Tools → test

Shims → fuzz.