IronFleet: Proving Practical Distributed Systems Correct

Hawblitzel et al.

Scriber: Haozhen Ding

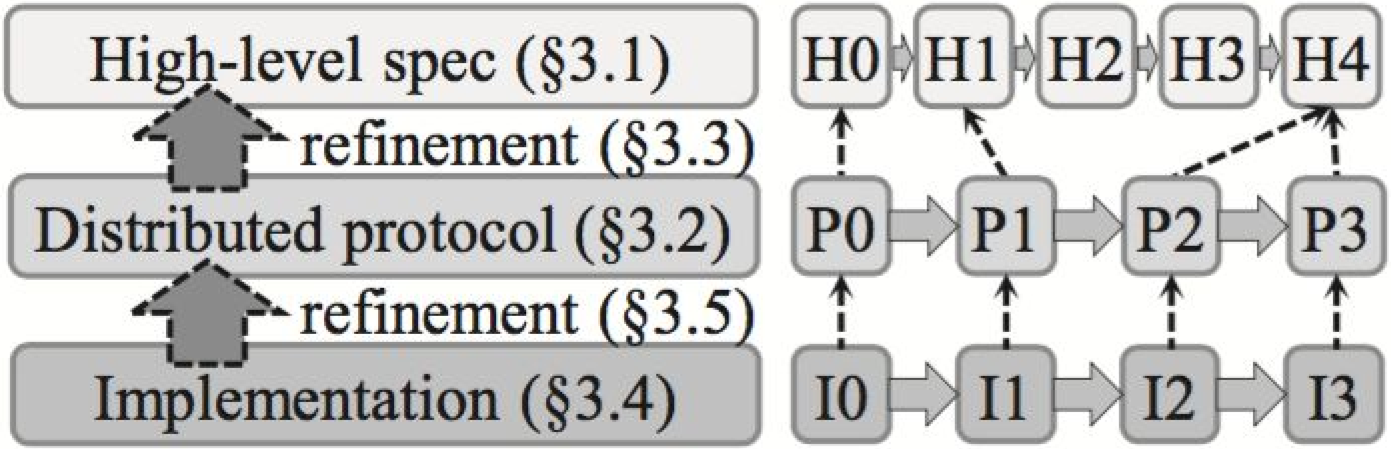
Recap

IronFleet

- Provable correctness of safety and liveness of distributed system implementation

Methodology

- Two-layer refinement



Recap

Methodology

- Floyd-Hoare verification (Dafny, Z3)
- Temporal Logic of Actions (TLA) (for liveness)

Techniques

- Always-enabled actions (for liveness)
- Concurrency containment via reduction
- Invariant quantifier hiding (constructive proof)
- etc.

Implementation/Evaluation

- IronRSL (replicated state-machine library)
- IronKV (sharded key-store)

Pros

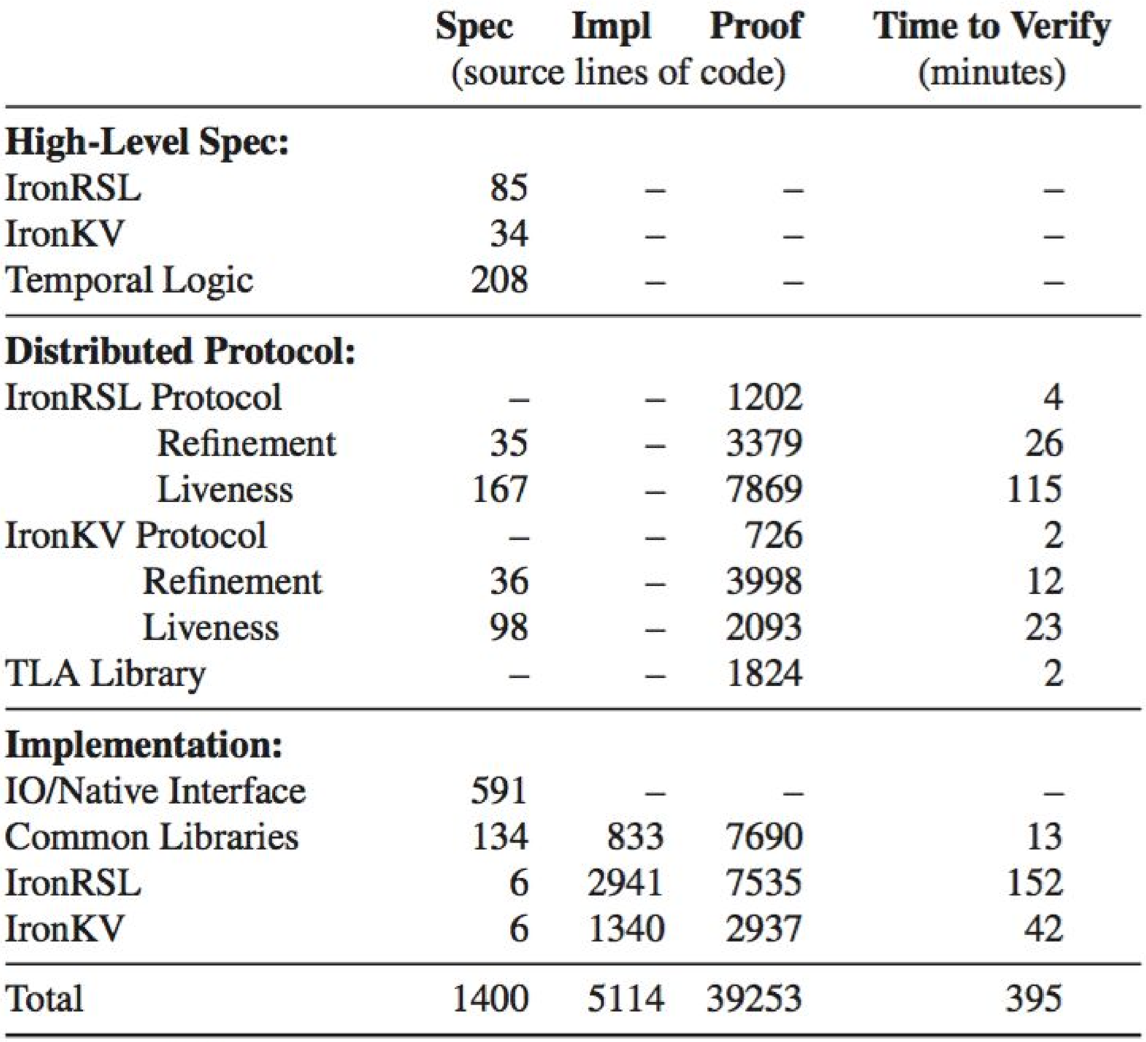
+ Formal guarantees
+ Both safety and liveness
+ Novelty in two-layer refinement
+ Two verified systems have comparable performance
+ Near-real-time IDE feedback
+ Libraries
+ Lesson learned section
+ Fair assumptions
    + Non-reliable network

Cons

- Much development effort
    - Proof code = 8x impl. Code
    - 3.7 person-years
- SMT solver complexity, need hints
- Dafny (or something similar)
- Compatibility with C++, Java?
- Hardness of heap management
- Exp. programs are CPU-bound
- Single threaded impl. on each host
- Formal proof of the atomicity reduction argument is future work

Discussion Questions

- IronFleet requires up to 8x lines of code for proof in additional to code yet achieves average performance. How do we balance the tradeoff between performance optimization and formal guarantee? **Is it worth the effort?**



	Spec	Impl	Proof	Time to Verify
	(source lines of code)			(minutes)
High-Level Spec:				
IronRSL	85	–	–	–
IronKV	34	–	–	–
Temporal Logic	208	–	–	–
Distributed Protocol:				
IronRSL Protocol	–	–	1202	4
Refinement	35	–	3379	26
Liveness	167	–	7869	115
IronKV Protocol	–	–	726	2
Refinement	36	–	3998	12
Liveness	98	–	2093	23
TLA Library	–	–	1824	2
Implementation:				
IO/Native Interface	591	–	–	–
Common Libraries	134	833	7690	13
IronRSL	6	2941	7535	152
IronKV	6	1340	2937	42
Total	1400	5114	39253	395

Figure 12. *Code sizes and verification times.*

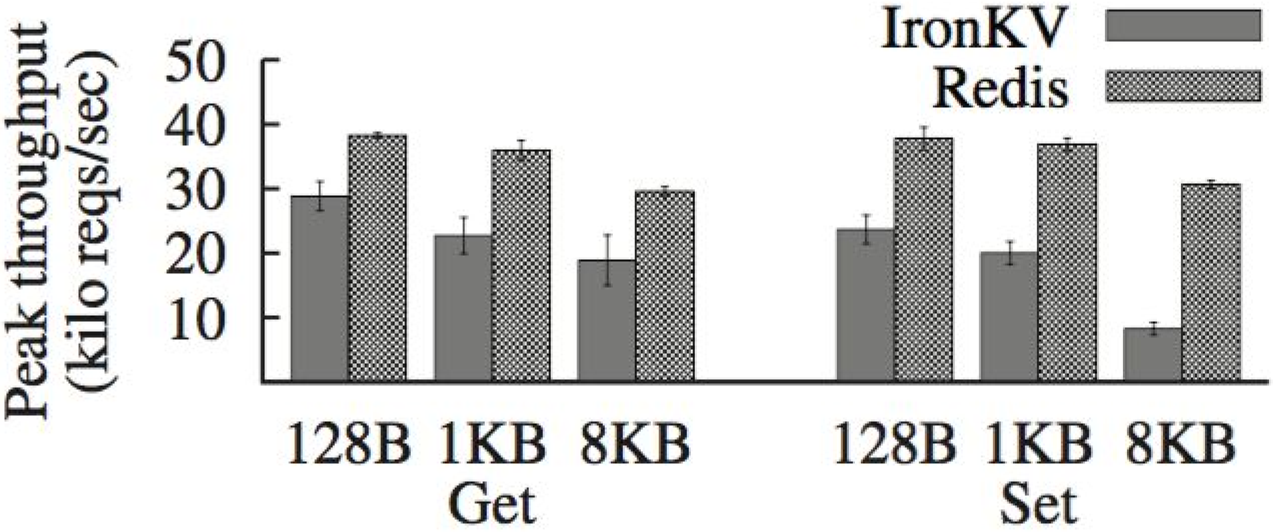


Figure 14. *IronKV's performance is competitive with Redis, an unverified key-value store. Results averaged over 3 trials.*

Discussion Questions

- IronFleet requires up to 8x lines of code for proof in additional to code yet achieves average performance. How do we balance the tradeoff between performance optimization and formal guarantee? **Is it worth the effort?**

  System requirement

  - Consistency vs availability
  - Failure recovery

  Business concern

Discussion Questions

- What are still in the protocol / implementation models assumed in IronFleet?

    - File storage?
    - Multi-threaded program?
    - Failure recovery?

Discussion Questions

- What are still missing in the protocol / implementation models assumed in IronFleet?

    - File storage? (memory)
    - Multi-threaded program? (not clear, additional proof)
    - Failure recovery? (part of distributed protocol)

Discussion Questions

- The paper proves Paxos liveness based on bounded message delay while in real network Paxos is not live. It might be that IronFleet verifies the correctness of a system but it is actually built upon unrealistic assumptions. How much can we trust our assumptions or the result of IronFleet?

Discussion Questions

- The paper proves Paxos liveness based on bounded message delay while in real network Paxos is not live. It might be that IronFleet verifies the correctness of a system but it is actually built upon unrealistic assumptions. How much can we trust our assumptions or the result of IronFleet?

  At least as much as we can trust them **without** verification.

Discussion Questions

- The paper proves Paxos liveness based on bounded message delay while in real network Paxos is not live. It might be that IronFleet verifies the correctness of a system but it is actually built upon unrealistic assumptions. How much can we trust our assumptions or the result of IronFleet?

  At least as much as we can trust them **without** verification.

- Is it bad to assume the correctness of hardware, OS, compilers, Dafny, etc?

Discussion Questions

- The paper proves Paxos liveness based on bounded message delay while in real network Paxos is not live. It might be that IronFleet verifies the correctness of a system but it is actually built upon unrealistic assumptions. How much can we trust our assumptions or the result of IronFleet?

  At least as much as we can trust them **without** verification.

- Is it bad to assume the correctness of hardware, OS, compilers, Dafny, etc?

  No. We need layers of abstraction.

Discussion Questions

- The entire IronFleet suit took 3.7 human-years to build. Can we cut the development time in the future?

Discussion Questions

- The entire IronFleet suit took 3.7 human-years to build. Can we cut the development time in the future?

  Certainly

  - More verified common libraries
  - Lessons learned about proof techniques
  - Incremental change to codebase may not need more proofs
  - Verification-aware development community

Discussion Questions

- Piazza: How comparable is IronFleet to Maude (from UIUC)?