IronFleet: Proving practical distributed systems correct
Hawblitzel, Howell, Kapritsos, Lorch, Parno, Roberts, Setty, Zill

Presented by Aniruddh Agarwal
Figure 1: Typical Figure 2 from Byzantine fault paper: Our network protocol

Buggy code update triggers Bing, Yahoo outage

Google blames software bug for Friday night Gmail outage
“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable”

Leslie Lamport
“Program testing can be used to show the presence of bugs, but never to show their absence!”

Edsger Dijkstra, *Notes on Structured Programming (EWD249)*
(Automated) formal verification

Prove guarantees about all possible behaviors of a system

- Testing can only make guarantees about behaviors and cases that are explicitly tested
- Historically hard to apply to practical centralized systems, let alone distributed systems
IronFleet

Key contributions

Formally-verified implementations of common distributed systems

IronRSL
(replicated-state-machine library)

IronKV
(sharded key-value store)
IronFleet

Key contributions

Formally-verified implementations of common distributed systems

Comparable to production systems in terms of features and performance

Demonstrates a methodology for verifying practical distributed systems with existing tools

First non-trivial systems to have machine-checked proofs of liveness
IronFleet
Assumptions and network model

Proofs are not absolute

Small amount of code (including spec) is “trusted”

Dafny, .NET compiler and runtime, underlying OS and hardware are assumed to be correct

Network may arbitrarily delay, drop or duplicate packets, but does not tamper with them

Additional assumptions are required for proving liveness (due to FLP)

   Enough hosts are alive for long enough and that the network eventually delivers enough messages
State machine refinement
State machine refinement
Sharded key-value store
State machine refinement

Sharded key-value store
Floyd-Hoare verification

Natural formalization of how we reason about imperative programs

Prove assertions of the form \( \{ P \} \ C \ \{ Q \} \)

- \( P, Q \) are predicates about program state
- \( C \) is a program statement
- "If \( P \) holds and we run \( C \), then \( Q \) holds"

Correctness of an entire program is proved by composing assertions
Floyd-Hoare verification

```java
method halve(x:int) returns (y:int)
    requires x > 0;
    ensures y < x;
{ y := x / 2; }
```
Methodology
State Machines

Define state machines for implementation and high-level spec

1. Define state space
2. Define a predicate which characterizes initial states
3. Define a predicate characterizing states with a transition between them

Define a refinement function from the implementation machine to the spec machine
Methodology

Two-level refinement

Spec
S0 → S1 → S2 → S3 → S4

Implementation
I0 → I1 → I2 → I3 → I4
Methodology

Two-level refinement

Spec

\[ S0 \rightarrow S1 \rightarrow S2 \rightarrow S3 \rightarrow S4 \]

Protocol

\[ P0 \rightarrow P1 \rightarrow P2 \rightarrow P3 \rightarrow P4 \]

Implementation

\[ I0 \rightarrow I1 \rightarrow I2 \rightarrow I3 \rightarrow I4 \]
Methodology
Reduction

```csharp
method Main()
{
    var s: ImplState;
    s := ImplInit();
    while (true) {
        s := EventHandler(s);
    }
}
```
Methodology

Reduction

Host A Step 1
Host A Step 2
Host A Step 3

Host B Step 1
Host B Step 2
Host B Step 3
Methodology

Reduction

Enforce that receives precede sends in event handler

Assume in proof that all host steps are atomic
# Evaluation

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

- **IronRSL**
- **Baseline**
- **IronRSL (Batch)**
- **Baseline (Batch)**

**Latency (ms)**

- Throughput (kilo reqs/s)

**Peak throughput (kilo reqs/sec)**

- 128B
- 1KB
- 8KB
- IronKV
- Redis
Conclusion

Novel methodology decouples system into multiple layers (spec, protocol and implementation)

Reasoning about protocol subtleties, concurrency and implementation complexity is localized

Refinement and reduction used to tie individually-feasible layers together

The methodology supports proofs of safety and liveness properties of practical distributed systems with performance and features competitive with unverified reference implementations
Thanks for listening!

THE CONSENSUS IS
THAT YOU'RE AWESOME
References

https://web.eecs.umich.edu/~manosk/assets/papers/ironfleet-sosp15.pdf

https://www.youtube.com/watch?v=KRmItIdCU-Y

https://jaylarch.net/static/publications/IronFleetSOSP2015.pptx

https://web.eecs.umich.edu/~manosk/summer-school-2020.html

https://twitter.com/borjasotomayor/status/995451438120697857