Compiling Distributed Systems with PGo

AS SEEN AT ASPLOS'23!

Distributed System Bugs Are Costly

Race condition caused 12 hour outage

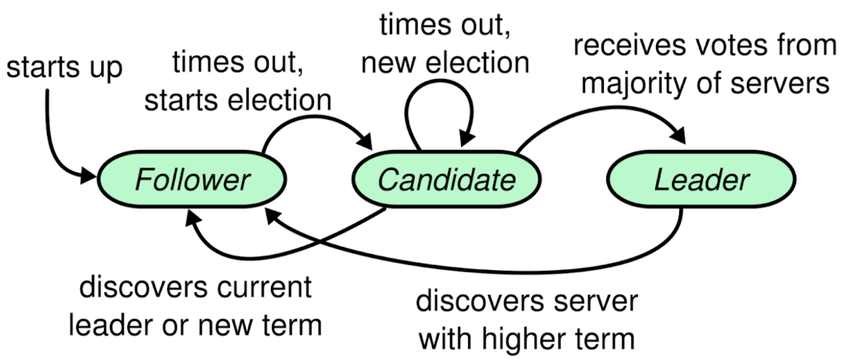
Misconfigured network caused connectivity loss

Infinite loop caused services ot hang

Formal Verification Helps Avoid Protocol Bugs

[1] [2] [3] [4]

... But Implementation Details May Not Match Protocol Description



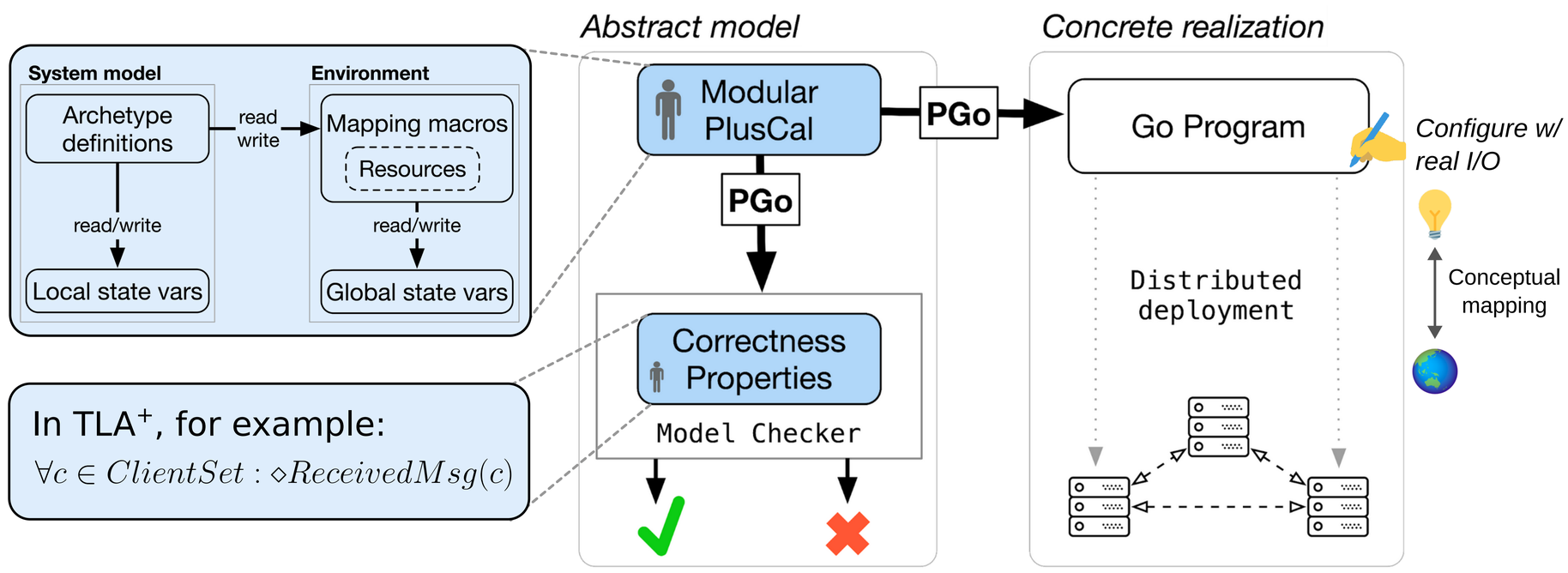
gap

```go
func (v *Buffers) Read(p []byte) (n int, err error) {
    for len(p) > 0 && len(*v) > 0 {
        n0 := copy(p, (*v)[0])
        v.consume(int64(n0))
        p = p[n0:]
        n += n0
    }
    if len(*v) == 0 {
        err = io.EOF
    }
    return
}

func (v *Buffers) consume(n int64) {
    for len(*v) > 0 {
        ln0 := int64(len((*v)[0]))
        if ln0 > n {
            (*v)[0] = (*v)[0][n:]
            return
        }
        n -= ln0
        (*v)[0] = nil
        *v = (*v)[1:]
    }
}
```

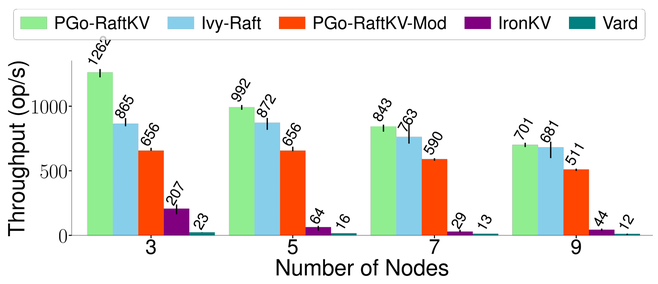
- ✗ Missing edge cases
- ✗ Ad-hoc error handling
- ✗ Hidden assumptions

Avoid Incorrect Implementations Using Compilation

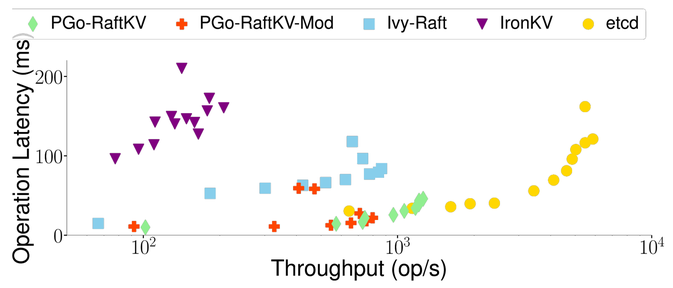


distcompiler.github.io

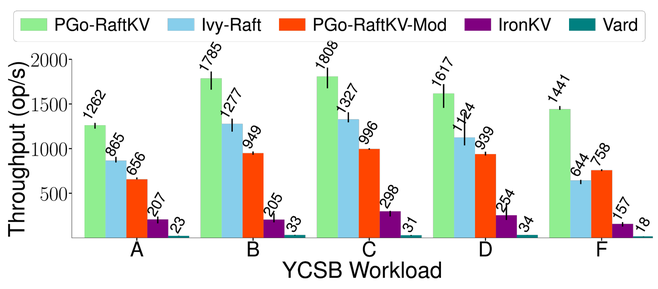
Eval: Raft-based Key-value Store (PGo-RaftKV)



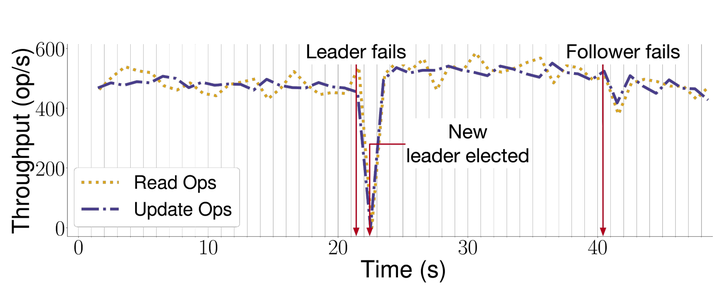
YCSB Throughput w/ Different Cluster Sizes



Latency / Throughput for YCSB Workload A



Peak Throughput for YCSB Workloads



Throughput During Failure / Recovery

Modular PlusCal Language

```
variables network = <<>>;

process (Server = 0) ==
  instance AServer(ref network[_], ...)
    mapping network[_] via TCPChannel

archetype AServer(ref network[_], ...)
...
readMessage:
  msg := network[self];

mapping macro TCPChannel {
 read {
    await Len($variable) > 0;
    with (msg = Head($variable)) {
      $variable := Tail($variable);
      yield msg;
    };
  }
  write {
    await Len($variable) < BUFFER_SIZE;
    yield Append($variable, $value);
  }
}
```


References

[1] Lamport, Leslie. "Specifying systems: the TLA+ language and tools for hardware and software engineers." (2002).
[2] Leino, K. Rustan M. "Dafny: An automatic program verifier for functional correctness." Logic for Programming, Artificial Intelligence, and Reasoning: 16th International Conference, LPAR-16, Dakar, Senegal, April 25–May 1, 2010, Revised Selected Papers 16. Springer Berlin Heidelberg, 2010.
[3] Padon, Oded, et al. "Ivy: safety verification by interactive generalization." Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2016.
[4] The Coq Development Team. The Coq Proof Assistant, version 8.9.0, 2019.

Finn Hackett, Shayan Hosseini, Renato Costa, Matthew Do, Ivan Beschastnikh

University of British Columbia

NSERC CRSNG