Runway

A new tool for distributed systems design

Diego Ongaro
Lead Software Engineer, Compute Infrastructure

@ongardie
https://runway.systems

Outline

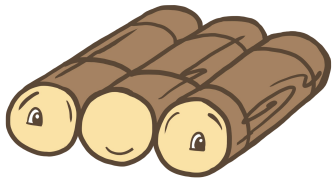
1. Why we need new tools for distributed systems design
2. Overview and demo of Runway
3. Building a Runway model

Distributed Systems Are Hard

- Concurrency and message delays
- Failures, failures during failures
- Many possible interleavings of events
- Little visibility, poor debugging environments

Raft Background / Difficult Bug

Raft: fault-tolerant consensus algorithm

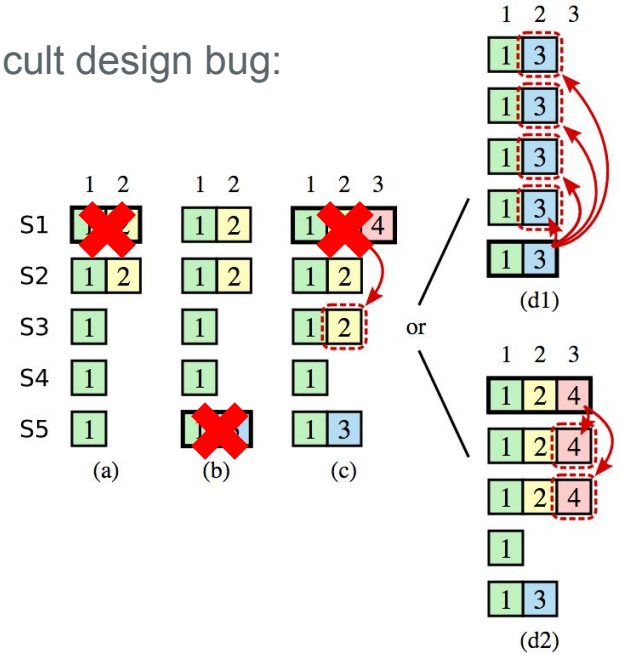


Used in many examples in this talk

Quick summary:

1. Use majority voting to elect a leader
2. Leader replicates its log to followers

Difficult design bug:



Typical Approaches Find Design Issues Too Late

Code reviews

Unit tests

System tests

Randomized tests, fuzzing, Jepsen

Benchmarks

Metrics

Dark launches

Bug reports

These are good techniques for implementation errors

- Localized: easy to fix

Too expensive for design errors

- May require large changes
- May cause unforeseen consequences

Let's find the right design sooner...

Design Phase

Goals

Communication:

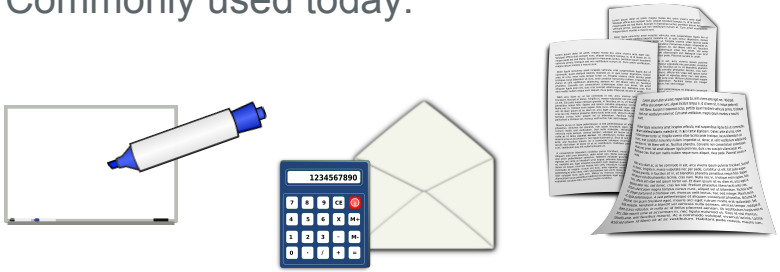
- Build intuition quickly
- Unambiguous
- Reviewable: discuss major issues and consider alternatives

Evaluation:

- Simplicity
- Correctness
- Performance
- Availability

Tools

Commonly used today:

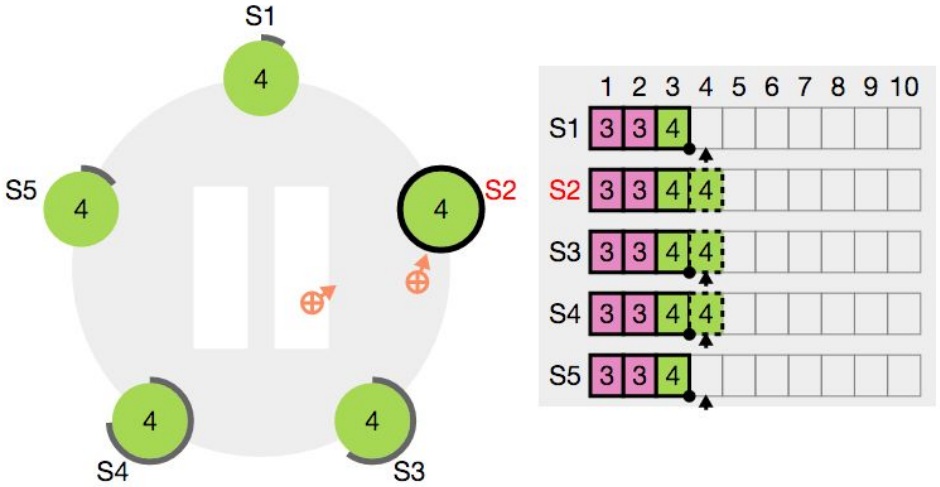


State of the art:

- Visualization (animation)
- Specification
- Model checking
- Simulation

Design Tools Use System Models

A model is a representation of a system that captures its essential concepts and omits irrelevant details.

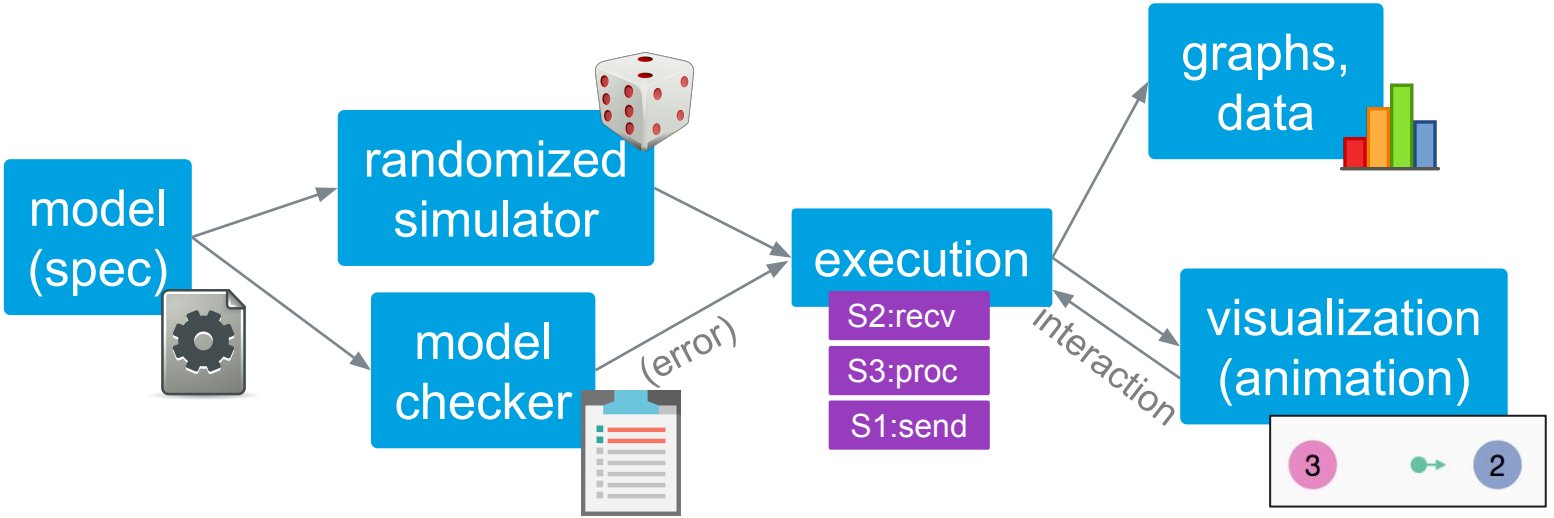


- Visualization
- Specification
- Model checking
- Simulation

A Tour of Runway

Runway Overview

Specify, simulate, visualize, and check system models



Integrated into one tool: write one model, get many benefits

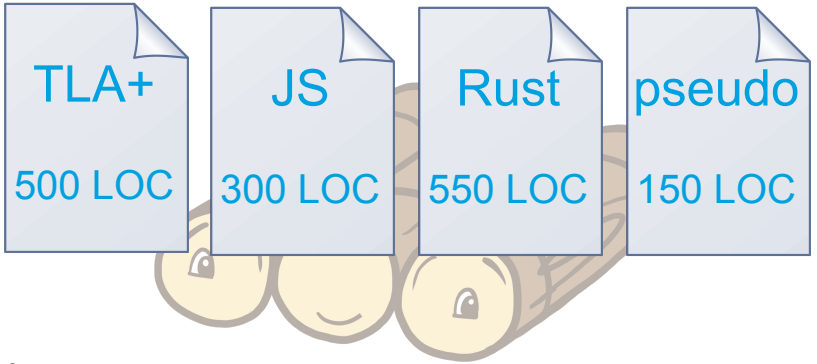
Runway Demo

Too many bananas, elevators, and Raft

Runway Integration

Independent tools: create independent models

- Write similar models for different tools
- Change the design: revise them all



Runway: reuse the same model

- Lower cost, additional benefit ⇒ create models sooner
- More likely to find modeling bugs

Specification, simulation, and model checking all benefit from visualization

Building a Runway Model

Developing a Model

specification

+

view

Idealized steps:

1. Sketch view by hand
2. Define types, state variables
3. Create view based on sketch
4. Write invariants
5. Write transition rules

visualization aids with debugging

Tip: set convenient starting state

Runway's Specification Language

- Specification is code
- Define starting state, transition rules, and invariants
  - Labeled Transition System
- Rules encode behavior + failures
- Applying a rule is atomic (one at a time)
- A rule is *active* if applying it would change the state
- If multiple rules are active, system decides
  - Simulator: random choice
  - Model checker: walk the tree

Example: Too Many Bananas (1)

Type and variable declarations, invariant

```
1  var bananas : 0..100;
2  var notePresent: Boolean;
3  type Person : either {
4  ··Happy,
5  ··Hungry,
6  ··GoingToStore,
7  ··ReturningFromStore {
8  ····carrying: 0..8
9  ··}
10 };
11 var roommates: Array<Person>[1..5];
```

type-safe variant: can't access unless ReturningFromStore

```
44 invariant BananaLimit {
45 ··assert bananas <= 8;
46 }
```

Example: Too Many Bananas (2)

Transition rule

```
13  rule step for state in roommates {
14      match state {
15          Happy {
16              state = Hungry;
17          }
18          Hungry {
19              if bananas == 0 {
20                  if (notePresent) {
21                      // Roommate at store:
22                  } else {
23                      notePresent = True;
24                      state = GoingToStore;
25                  }
26              } else {
27                  bananas -= 1;
28                  state = Happy;
29              }
30          }
31          GoingToStore {
32              state = ReturningFromStore {
33                  carrying: urandomRange(0, 8)
34              };
35          }
            ReturningFromStore(bag) {
                bananas += bag.carrying;
                notePresent = False;
                state = Hungry;
            }
41      }
42  }
```

no state changed: inactive until readset changes

It’s About Time

Developers: each server tries to approximate “the global clock”

Physicists: Ha! Blah blah blah, blah, blah! Blah blah blah blah. Blah!

Want some safety properties to hold even if clocks misbehave

Need time to describe availability and performance

Runway’s current approach: global clock, conditionally

```
rule startNewElection for serverId, server in servers {
    if past(server.timeoutAt) {
```

true

server.timeoutAt <= clock

Summary

- Let’s apply tools to help us design distributed systems
- Modeling helps focus our attention on concepts, leaving out unimportant details
- Runway combines spec, model checking, simulation, and interactive visualization
- Go view the models, build your own, and help develop Runway

solve design problems in the design phase

https://runway.systems