



# Virtual Machine Support for Debugging Concurrency Bugs

**No Distribution!**

Deterministic Replay, Asynchronous Snapshots,  
and Bug Detection for Event Loop Systems

**Stefan Marr**

Shonan, May 2019

Ongoing work with Dominik Aumayr, Carmen Torres Lopez,  
Elisa Gonzalez Boix, Hanspeter Mössenböck

# Ongoing work



involving



Carmen  
Torres Lopez



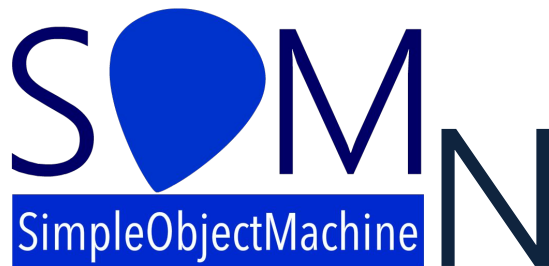
Elisa  
Gonzalez Boix



Dominik  
Aumayr



Hanspeter  
Mössenböck



Based on **GraalVM**<sup>TM</sup>

# Concurrency Bugs are Common in Event Loop Systems



53 projects, 57 issues

2 studies

12 projects, 1000 potential issues



12 projects

1 study

53 concurrency issues



Websites in top 500

≈1-10 concurrency issues



8-27 apps

≈2-20 concurrency issues



6 projects

35 known event races



**Tip of the  
Iceberg**

# Challenges

## bug reproduction

e.g. fails 1 in 10 times



## asynchronous snapshots

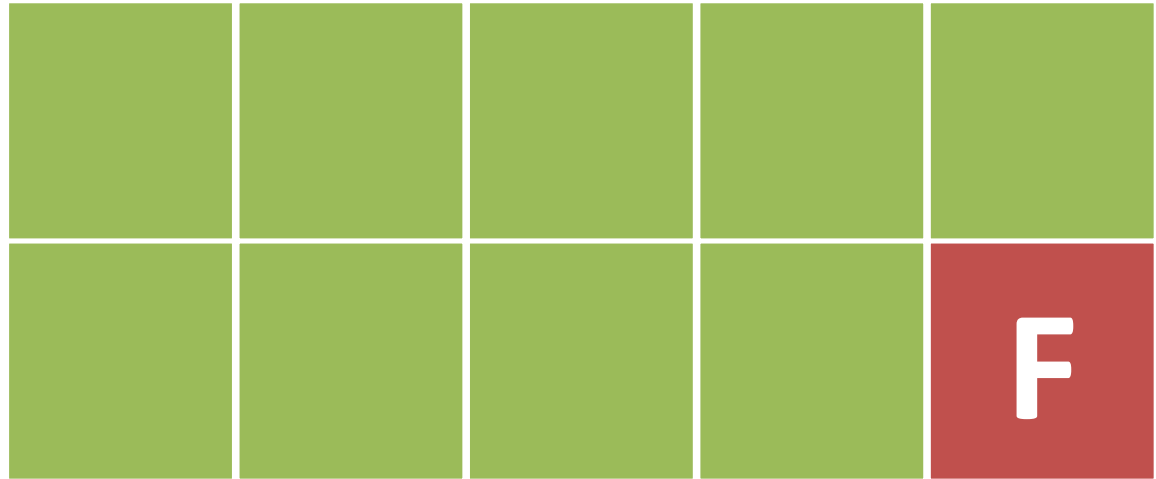
saving the world, without stopping it



## bug mitigation

fails 1 in 10 times, can we avert failure?



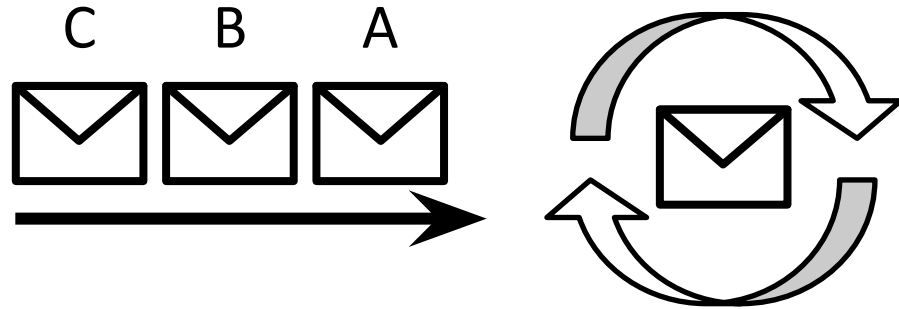


Reproduces only 1 in 10? How can I fix such a bug???

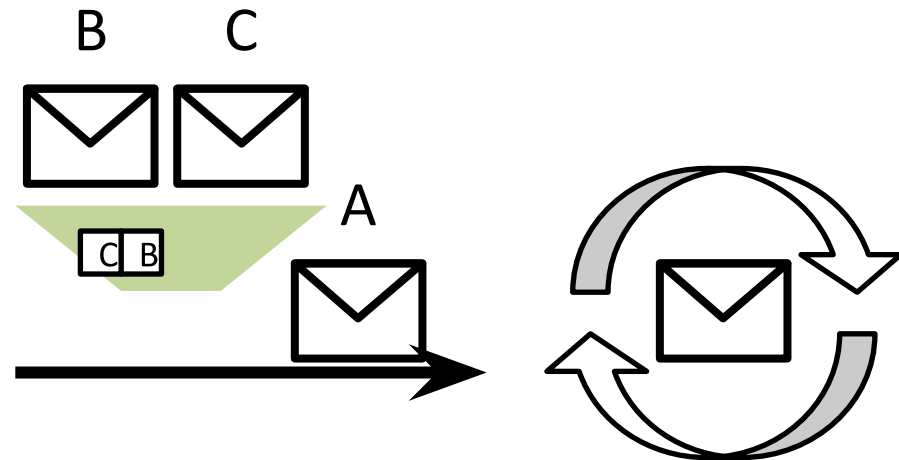
# **NON-DETERMINISM MAKES FOR UNHAPPY DEBUGGERS**

# One Solution

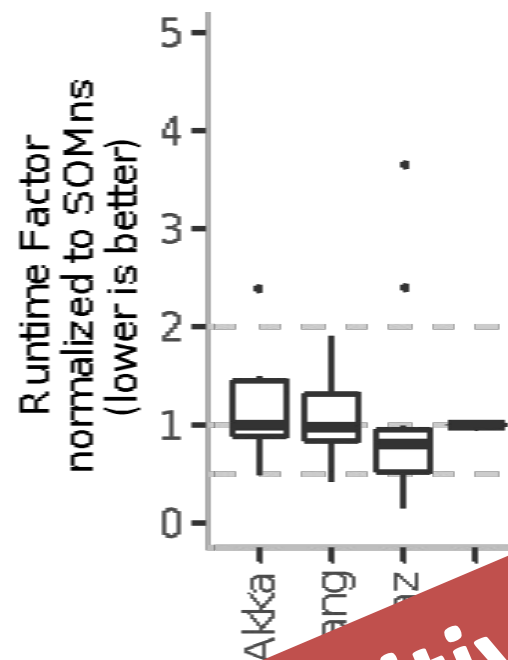
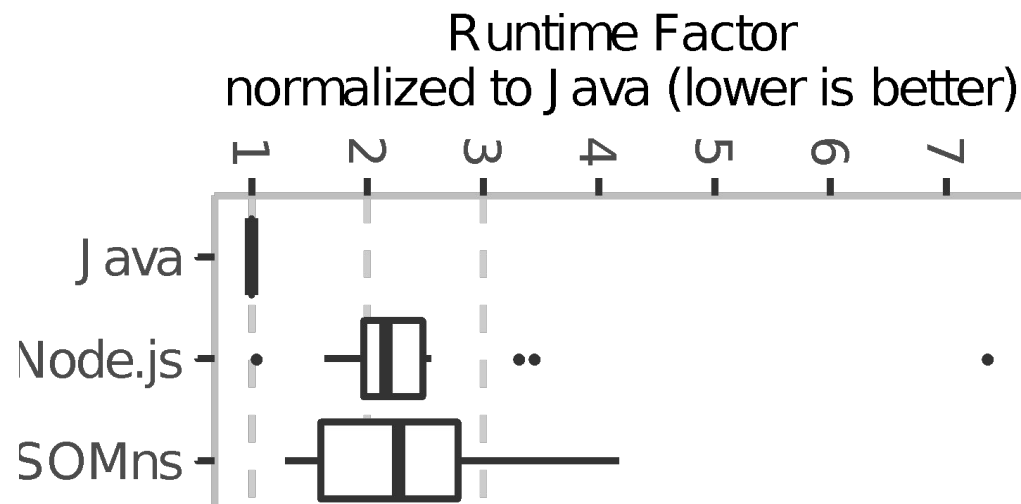
- Record event order



- Replay reorder to fit



# Performance: Baselines



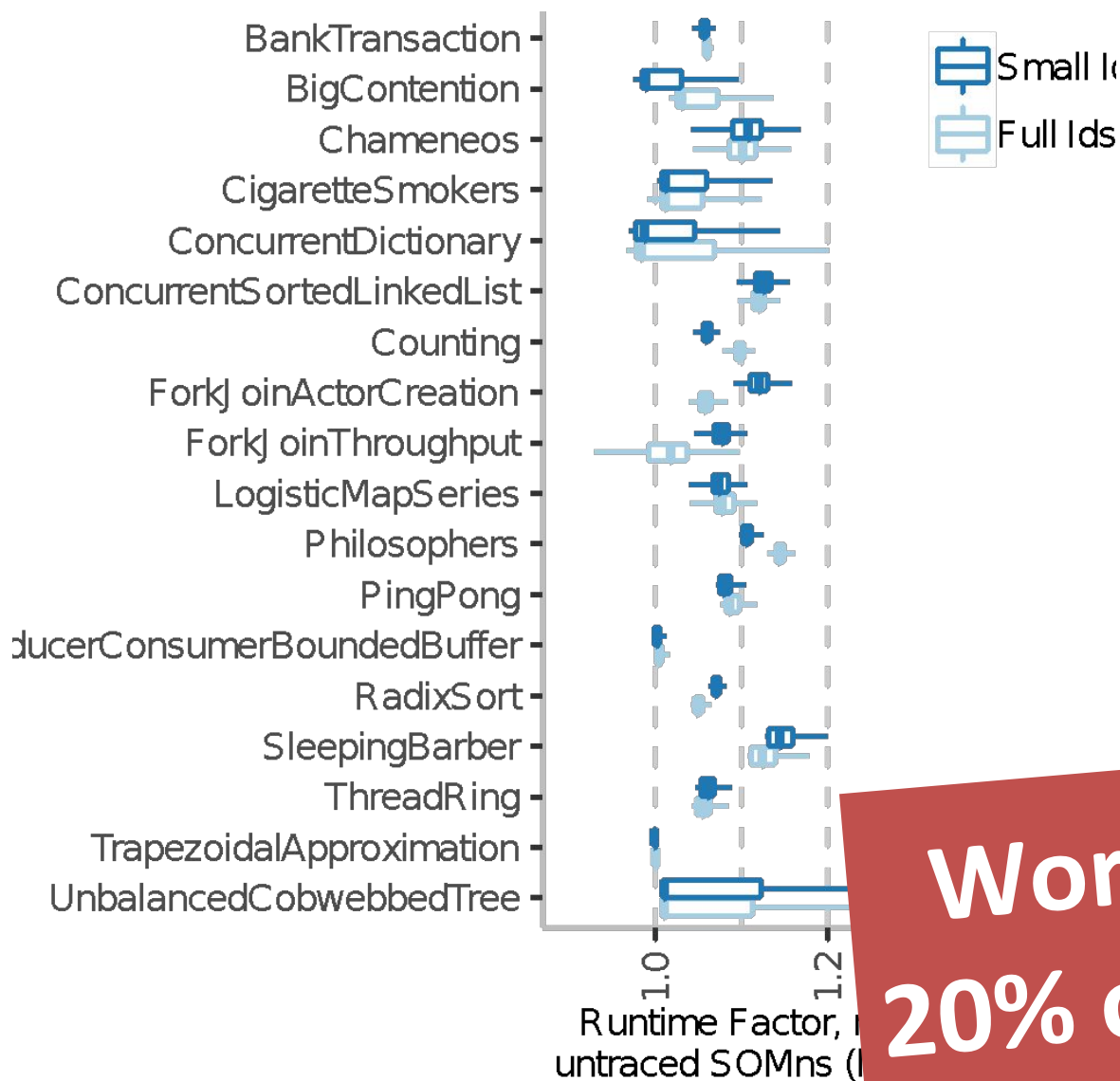
**On level of  
Dynamic  
Languages!**

Cross-Language C  
m/smarr/are-we-f  
chmark Suite  
om/shamsimam/savin

**Competitive  
with JVM  
frameworks!**



# Tracing Performance: Savina



**Worst case:  
20% overhead**





# AcmeAir: Microservices Evaluator

- Airline booking system
- Designed as benchmark

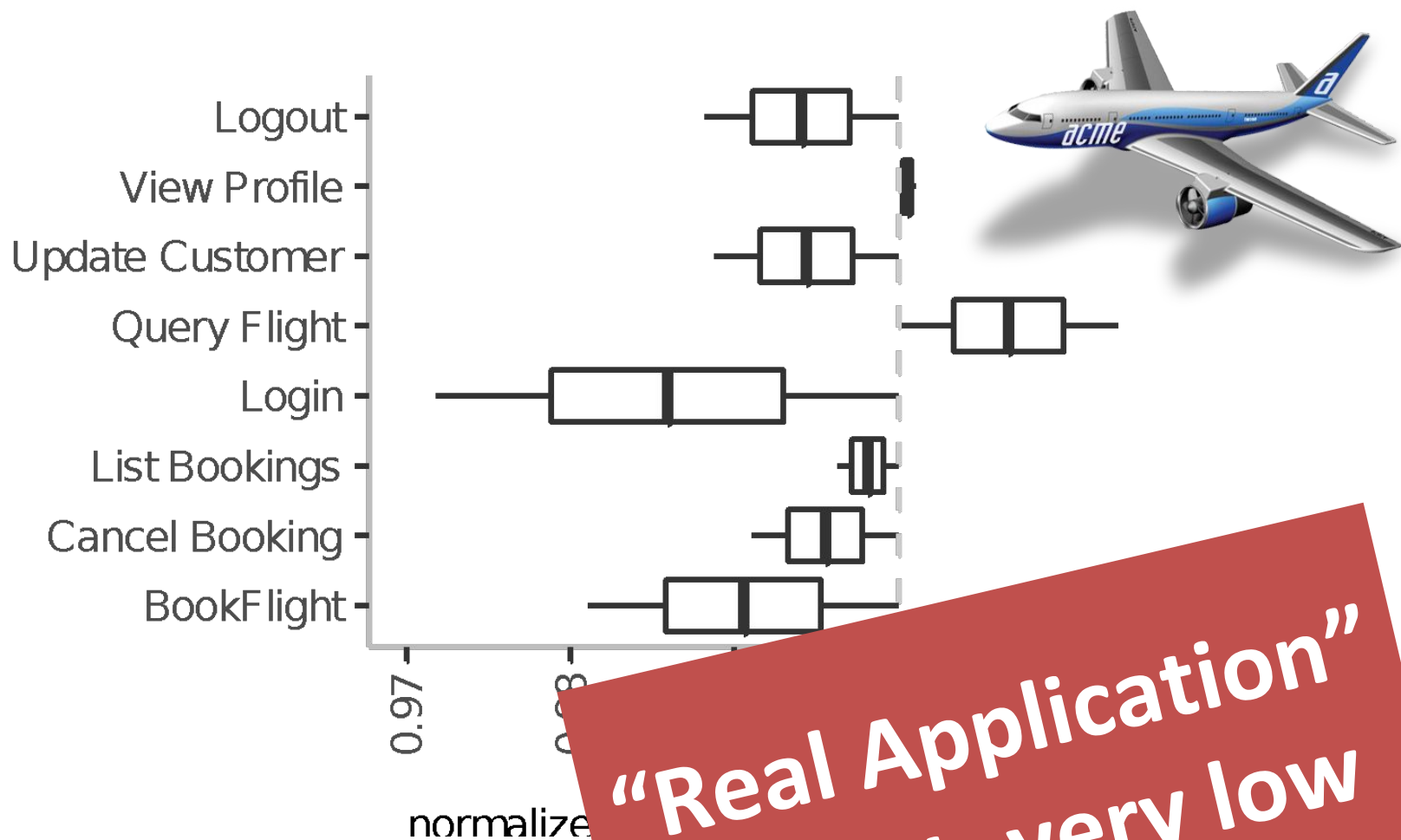
IBM



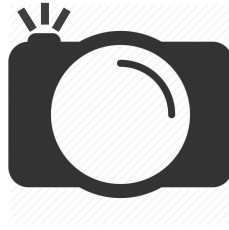
 <https://github.com/acmeair>



# Tracing Performance: Acme Air



**“Real Application”  
impact: very low**

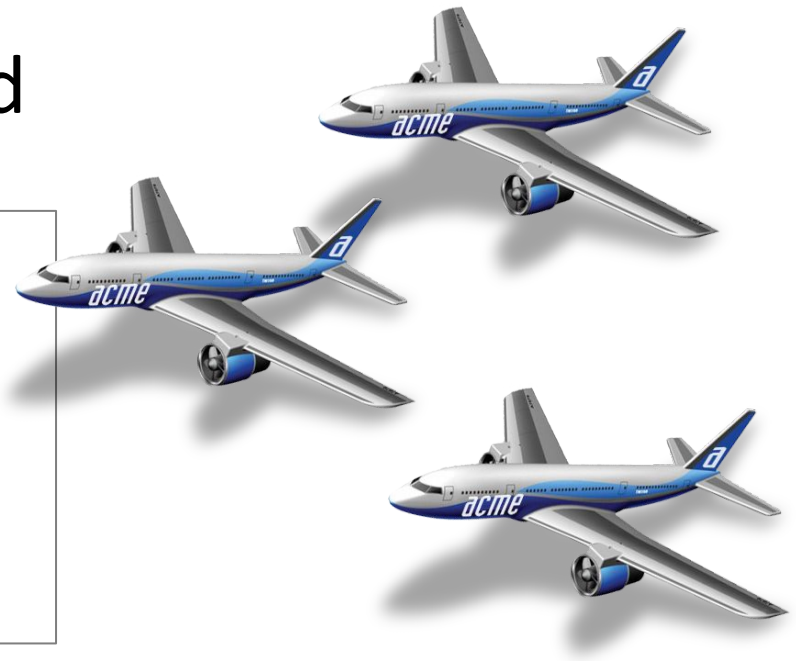
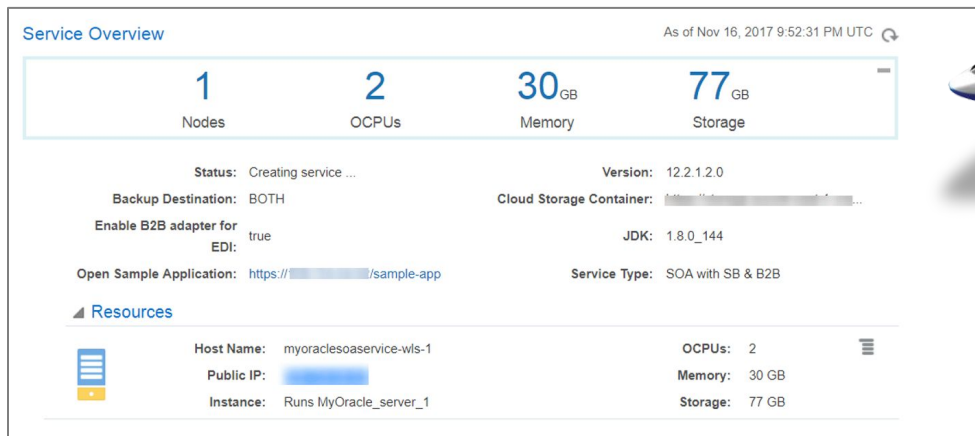


Snapshotting Actor Systems without Stopping Them

**LONG AND HUGE TRACES MAKE  
REPLAY IMPRACTICAL**

# Deterministic Replay and Snapshots

## Microservices in the Cloud



Service runs for days or weeks  
Can't have replay from start → Need Snapshots

# Deterministic Replay and Snapshots

## Microservices in the Cloud



### Service Overview

As of Nov 16, 2017 9:52:31 PM UTC

1	2	30 GB	77 GB
Nodes	OCPUs	Memory	Storage

Status: Creating service ... Version: 12.2.1.2.0

Backup Destination: BOTH Cloud Storage Container: [REDACTED]

Enable B2B adapter for EDI: true JDK: 1.8.0\_144

Open Sample Application: [https://\[REDACTED\]/sample-a](https://[REDACTED]/sample-a)

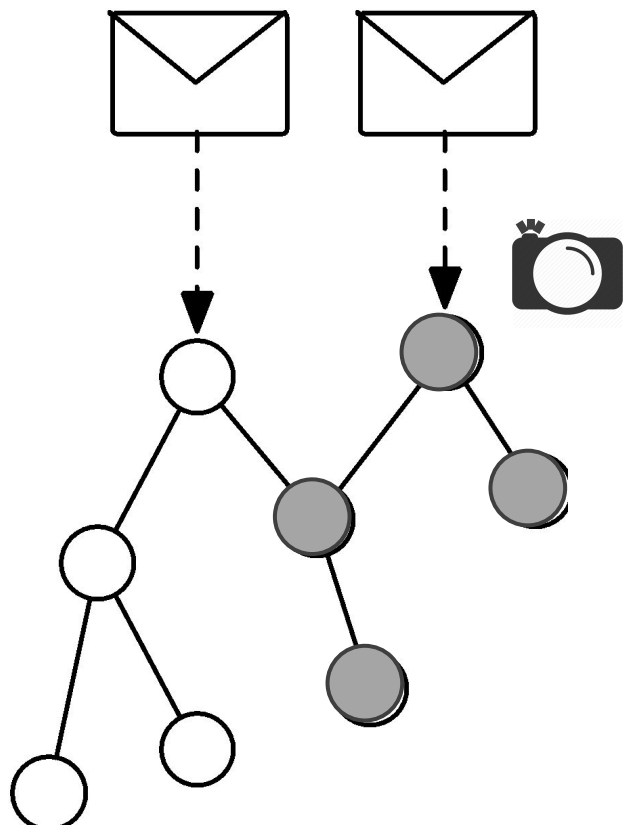
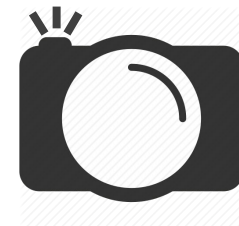
### Resources

	Host Name: myoraclesoaservice-wls-
	Public IP: [REDACTED]
	Instance: Runs MyOracle_server_1

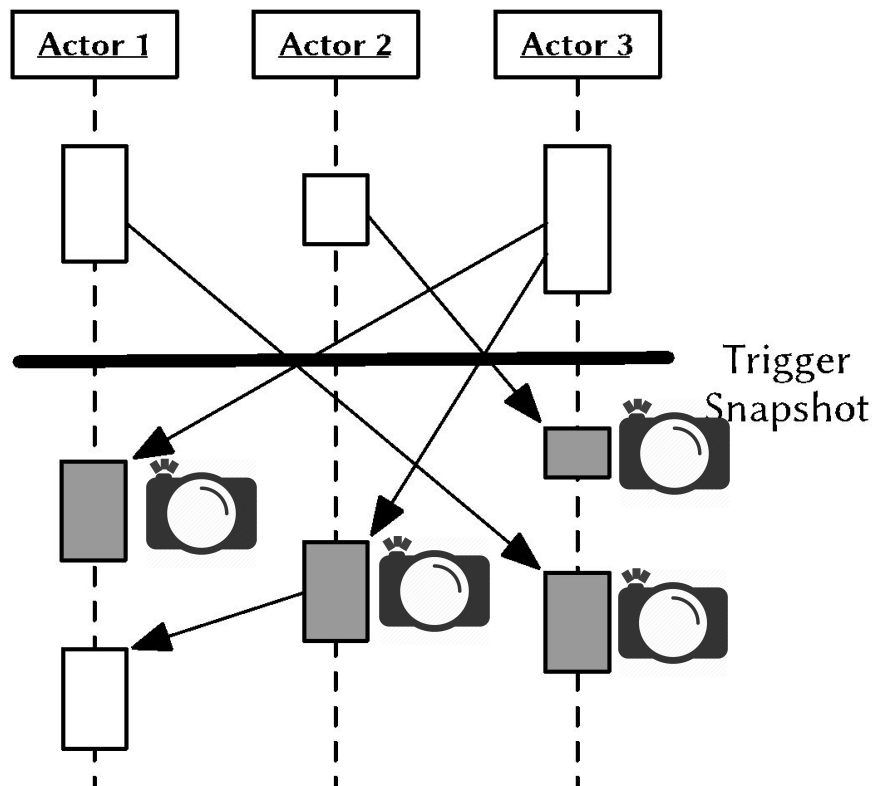
**Download Snapshot.  
Debug Locally!**



# Asynchronous and Partial Heap Snapshots



snapshot only objects  
reachable from a message



serialize partial heap  
before message execution



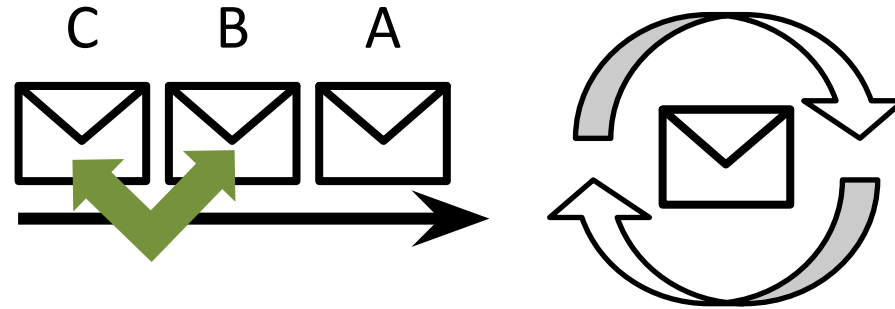
Where to go from here?

**THE FUTURE?**



# Bug Mitigation?

## Detect Event Races At Run Time



Order A -> B -> C problematic?

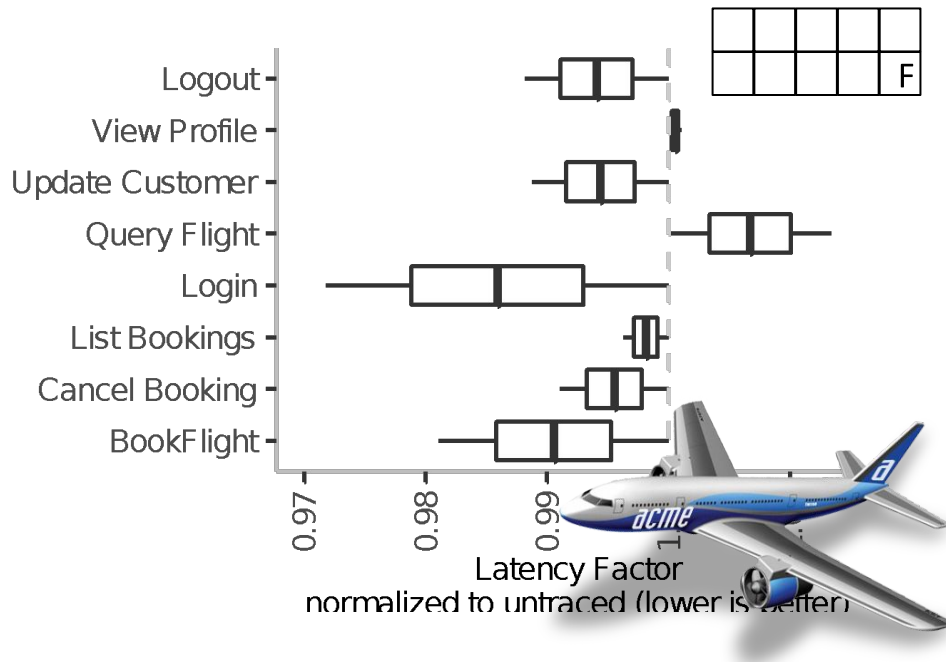
Let's swap them!



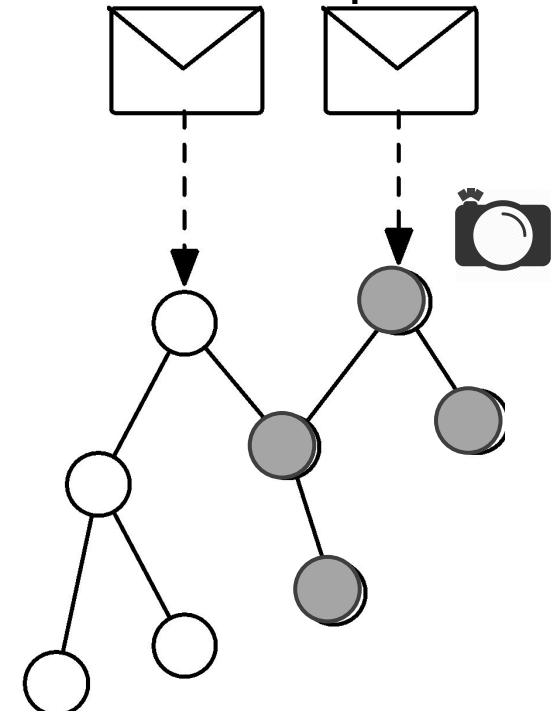
# Summary



Deterministic Replay  
with low overhead



Asynchronous  
Partial Snapshots



Future: Race Detection and Bug Mitigation

