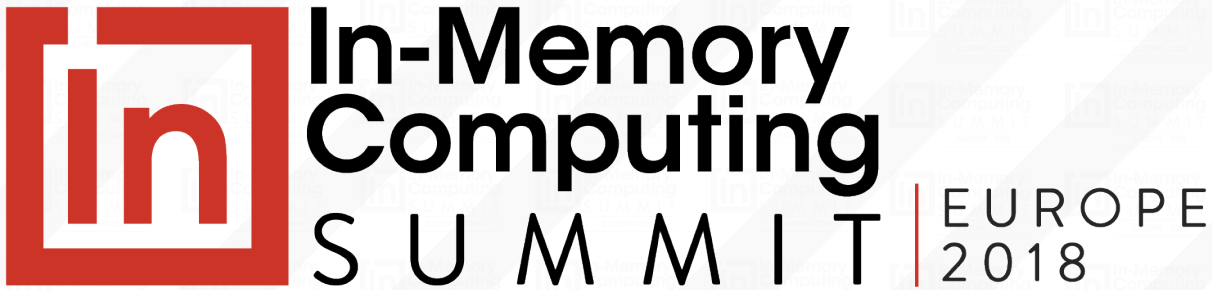


EXACTLY ONCE STATEFUL STREAMS THE EASY WAY

COLIN MACNAUGTHON

NEEVE RESEACH

INTRODUCTIONS

- Based in Silicon Valley
- Creators of the X Platform™ - Memory Oriented Application Platform.
- Passionate about high performance computing for mission critical enterprises.

WHY DO WE CARE ABOUT STREAMING?

WHY STREAMING?

Loosely coupled, multi-agent micro services architectures are more agile, and reduce delivery risk. Coupled with the increasing amount of business valuable data it is important that we can move data between processes rapidly while at the same time maximizing hardware utilization to reduce cost.

WHY EXACTLY ONCE?

Reliability coupled with ease: the less developers have to focus on handling loss and duplicates the more robust our multi agent applications will be.

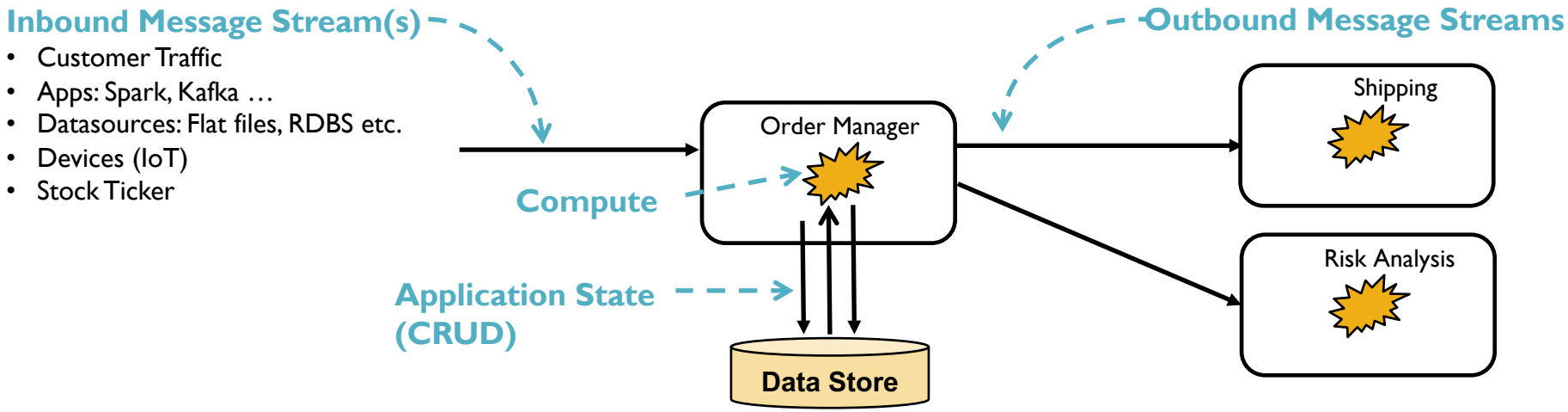
AGENDA

- Why is Exactly Once Streaming Hard?
- How The X Platform tackles Streaming
- Streaming Usecase: IoT Fleet Tracking

STREAM TRANSACTION PROCESSING APPLICATIONS

What do they do?

1. Consume Inbound Messages
2. Read / Update State
3. … and Produce Outbound Messages



THE IDEAL STREAMING FRAMEWORK

- **Fast -** 10s - 100k transactions/sec, response times in microseconds or milliseconds
- **Stateful –**Ability to operate on persistent state in a transactionally consistent fashion.
- **Reliable -** no dups / no loss / atomic across failures
- **Available –** handle process / infrastructure failures
- **Scalable -** scale on demand
- **Manageable -** integrate with CI (test, build, provision)
- **Easy -** trivial to author and drop in new stream processors *without concern for the above.*

MICROSECONDS MATTER

A processing time of 1ms limits your throughput to 1000 messages / sec.

Same applies to any synchronous callouts in the stream.

To achieve >10k Transactions/Second you must leverage In Memory technologies

MICROSECONDS MATTER

Storage	Latency	Ops/Sec
L1 Cache	~1ns	1b
L2 Cache	~3ns	333m
L3 Cache	~12ns	83m
Remote NUMA Node	~40ns	25m
Main Memory	~100ns	10m
Network Read	100µs	10k
Random SSD Read 4K	150µs	6.6k
Data Center Read	500µs*	2k
Mechanical Disk Seek	10ms	100

MEMORY ORIENTED COMPUTING!

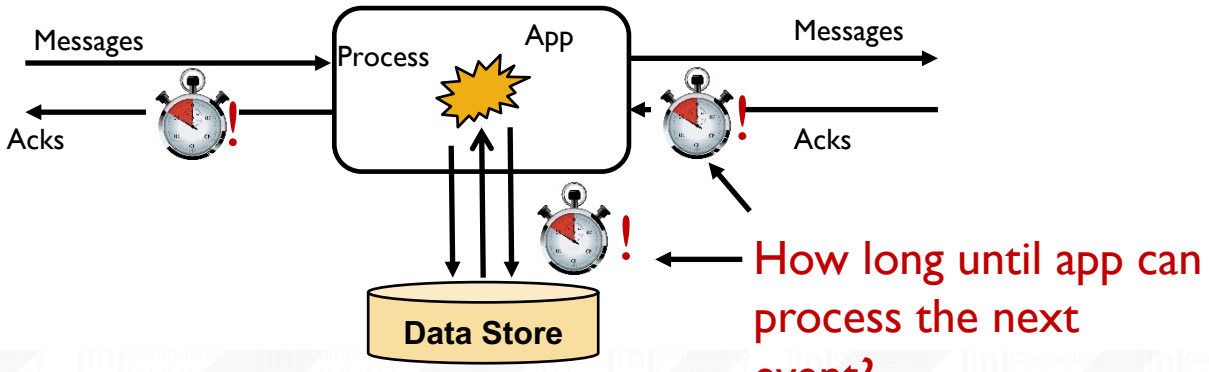
All State in Memory All The Time!

Non Starters For Performance We're Talking About!

Sources: https://gist.github.com/jboner/2841832
http://mechanical-sympathy.blogspot.com/2013/02/cpu-cache-flushing-fallacy.html

THE CHALLENGES

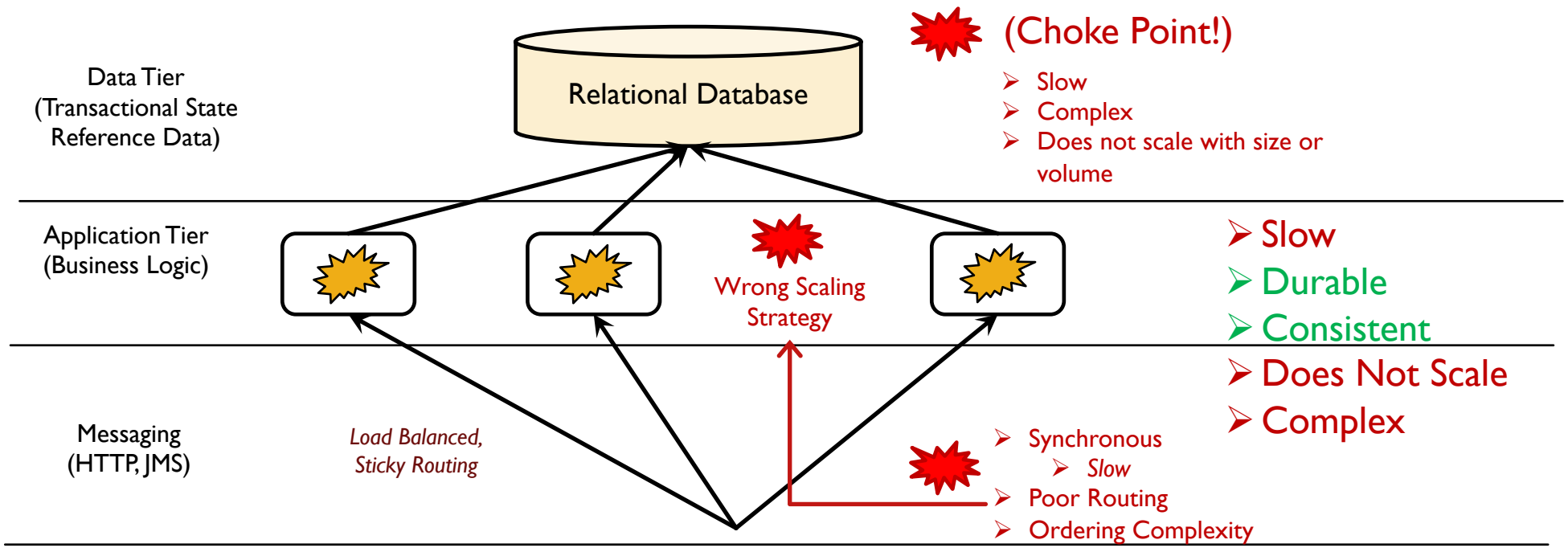
- Exactly Once Semantics
  - Messaging – No Loss / No Dups
  - Storage and Access to State – No Loss / No Dups
- Atomicity between Message Streams and Data/Stream Stream
  - Receive-Process-Send must be atomic for event processing consistency across failures.



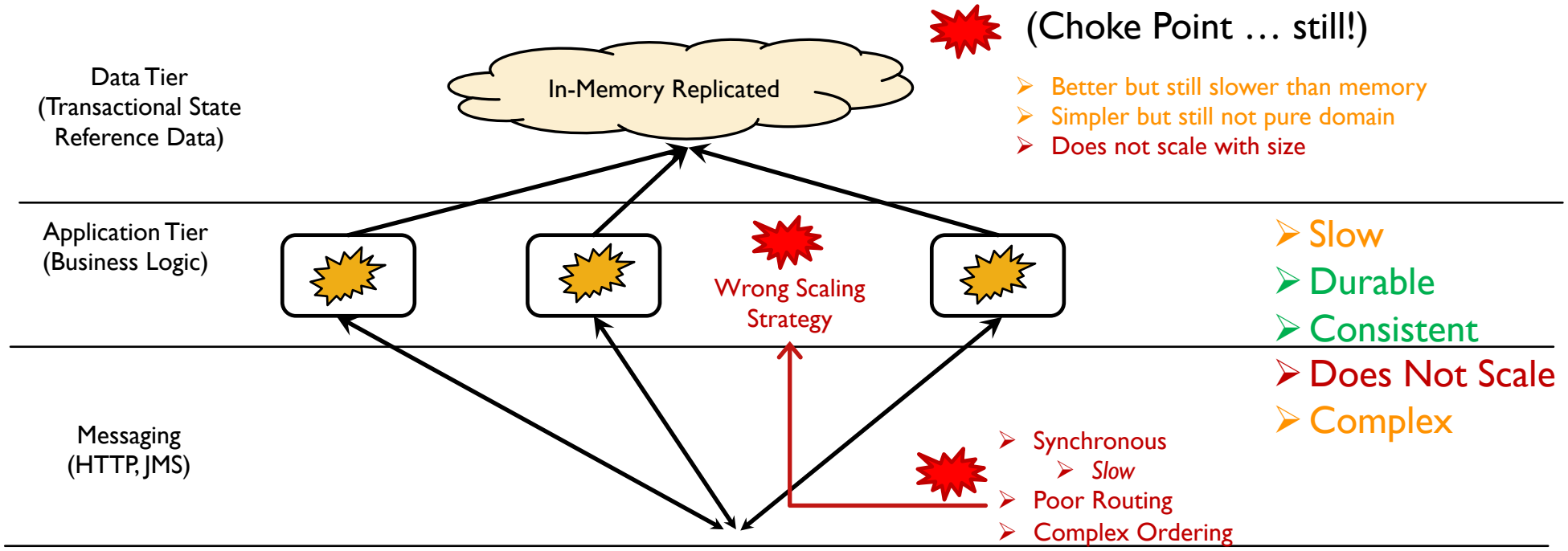
Storage is key - must remember:

- What events have already been processed
- Changes in state as a result of processing
- What results have (and have not) been sent to the world.

TRADITIONAL TP APPLICATION ARCHITECTURE

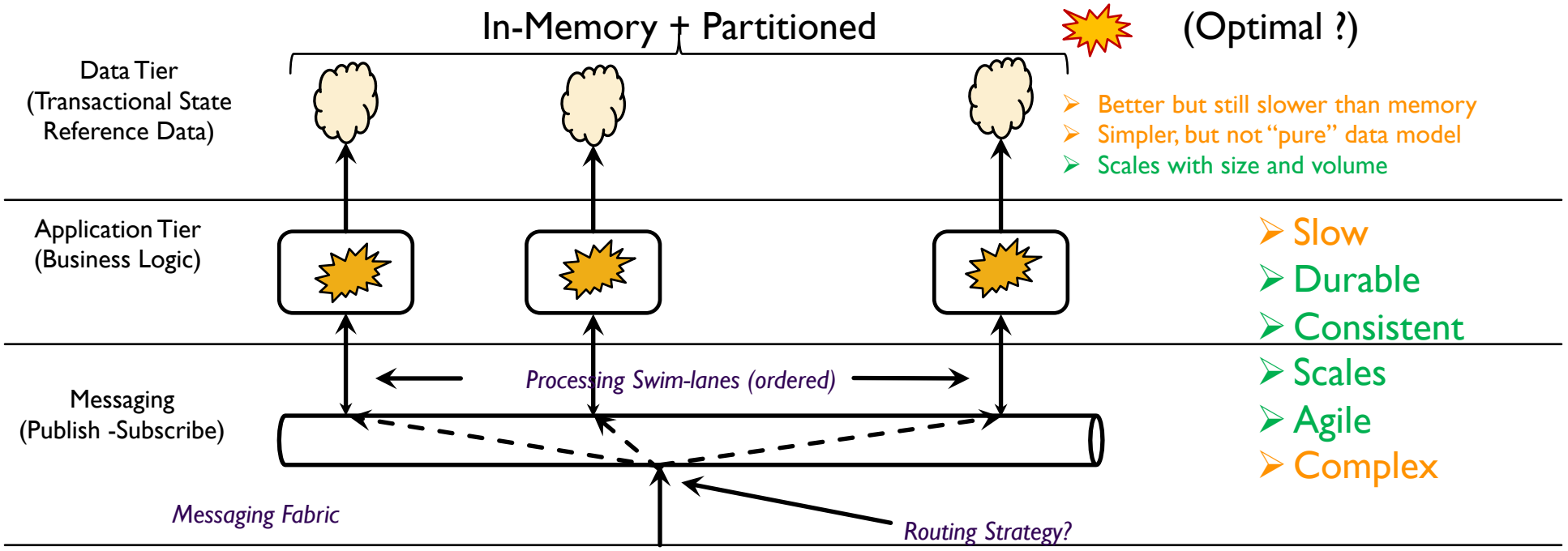


LAUNCH DATA INTO MEMORY



DATA GRAVITY
(DATA STRIPING + SMART ROUTING)

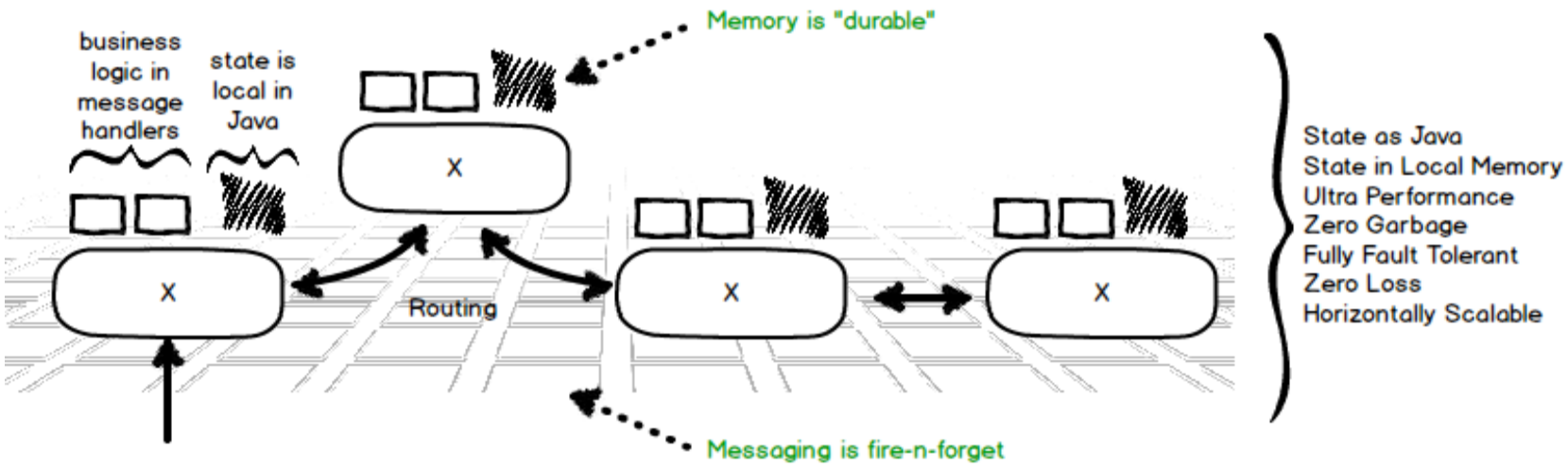
A MICRO SERVICE ARCHITECTURE



WHY STILL SLOW AND COMPLEX

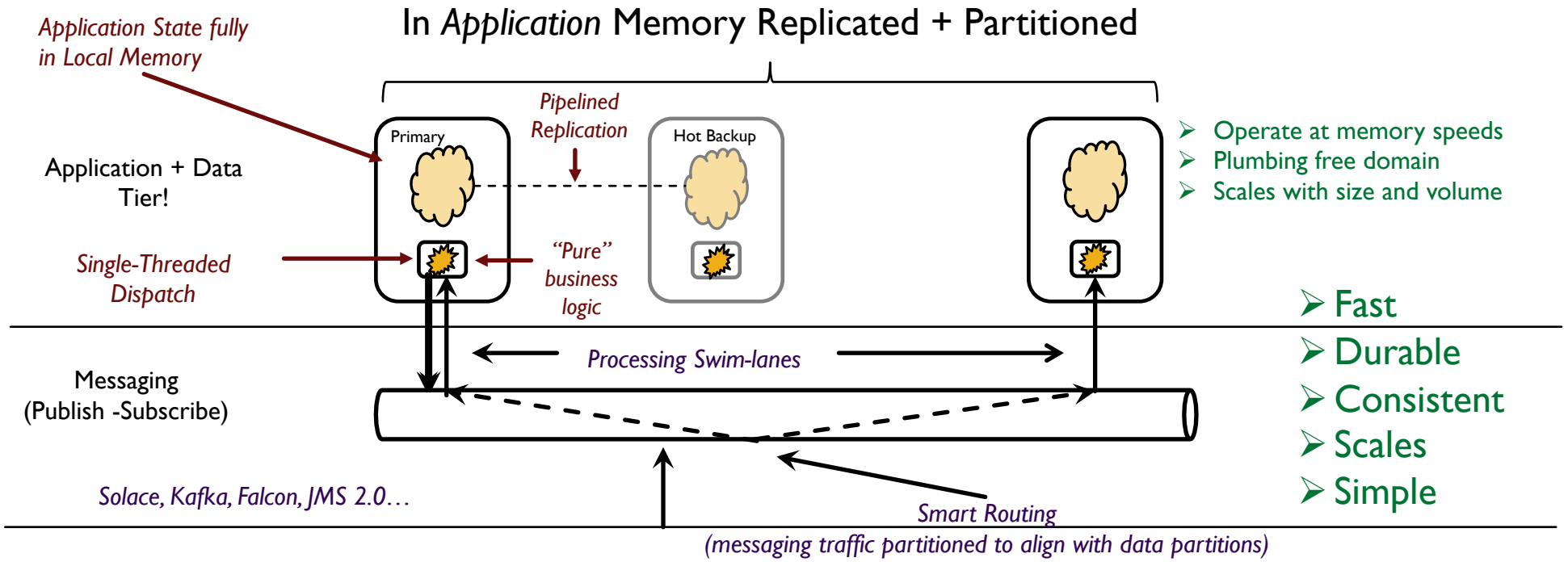
- **How Slow?**
  - Latency
    - 10s to 100s of milliseconds
  - Throughput
    - Not great with single pipe
    - Few 1000s per second per partitioning
- **Why Still Slow?**
  - Remoting out of process (data latency)
  - Synchronous data updates and message acknowledgement
  - Concurrent transactions are not cheap!
- **Why Complex?**
  - Transaction Management still in business logic
  - Thread management for concurrency (only way to scale)
  - Complex Routing (how to load balance between swim lanes?)
  - Data transformations due to lack of structured data models

STREAMING APPS ON THE X PLATFORM



- ✓ **Message Driven**
- ✓ **Stateful**
- ✓ **Multi-Agent**
- ✓ **Totally Available**
- ✓ **Horizontally Scalable**
- ✓ **Ultra Performant**

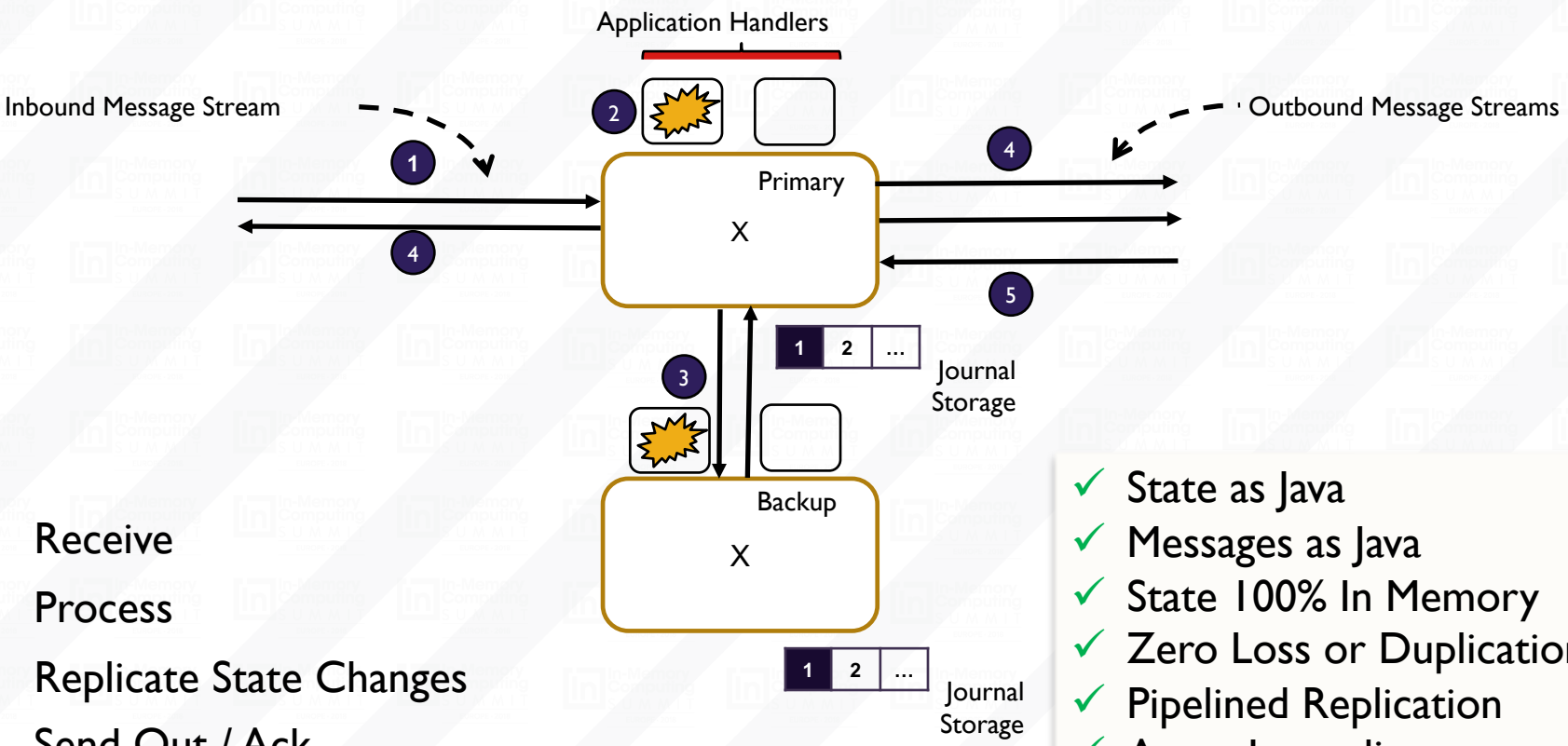
THE X PLATFORM APPROACH



NOW WHAT IS THE PERFORMANCE?

- How Fast?
  - Latency
    - 10s of microseconds to low milliseconds
  - Throughput
    - 100s of thousands of transactions per second
- How Easy?
  - Model Objects and State in XML, generated into Java objects and collections.
  - Annotate methods as event handlers for message types.
  - Single threaded processing
  - Work with state objects treating memory as durable.
  - Send outbound messages as “Fire And Forget”
  - Shard applications by state, messages routed to right app.

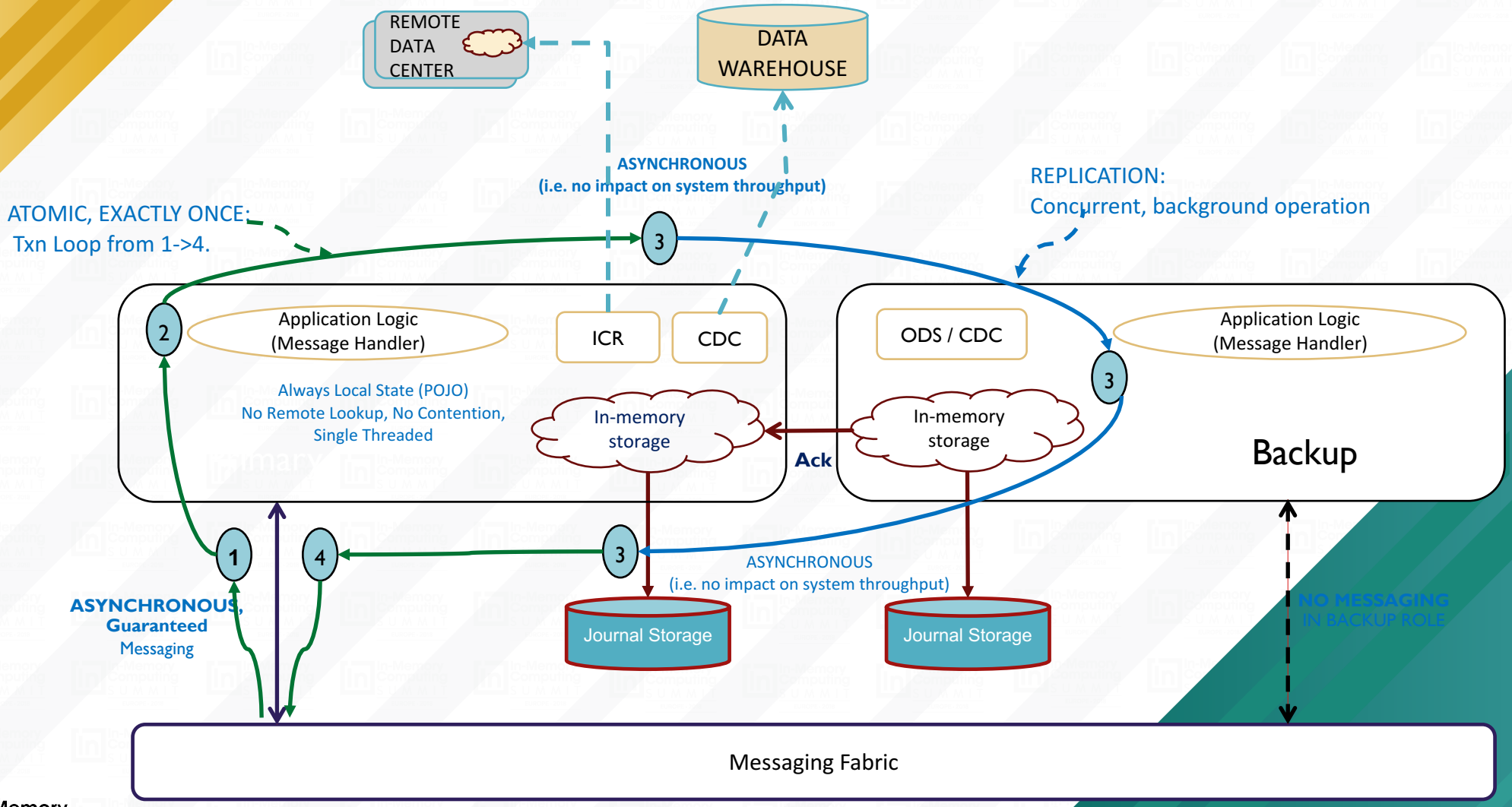
X PLATFORM TRANSACTION PIPELINING (HA)



1. Receive
2. Process
3. Replicate State Changes
4. Send Out / Ack
5. Inbound Acks

- ✓ State as Java
- ✓ Messages as Java
- ✓ State 100% In Memory
- ✓ Zero Loss or Duplication
- ✓ Pipelined Replication
- ✓ Async Journaling
- ✓ Pipelined Messaging
- ✓ Pooling for Zero Garbage

THE FULL HA PICTURE



DEVELOPER CONCERNS

X Application

=

MESSAGES

+

STATE

+

BUSINESS LOGIC (HANDLERS)

```
EventHandler
final public void onAuthRequest(AuthRequestMessage message
                                 Repository state) {
    // instantiate a new cc transaction
    final Transaction txn = Transaction.create();

    // extract from message into a transaction
    AuthRequestMessageExtractor.extract(message, txn);

    // update transaction state
    txn.setState(TransactionState.PendingAuth);

    Customer customer = state.getCustomers().get(txn.getCustomerId()
    customer.getTransactions().add(txn)

    // create a fraud detection request
    final FraudDetectionRequest req = FraudDetectionRequest.create();

    // populate the request
    FraudDetectionRequestPopulator.populate(req, txn);

    // send the event
    sendMessage(req);
}
```

JAVA

+

CONFIG

Not required for vertical specific models such as FIX

Not required for Event Sourcing

USE CASE - IOT

Building a Fleet Tracking System

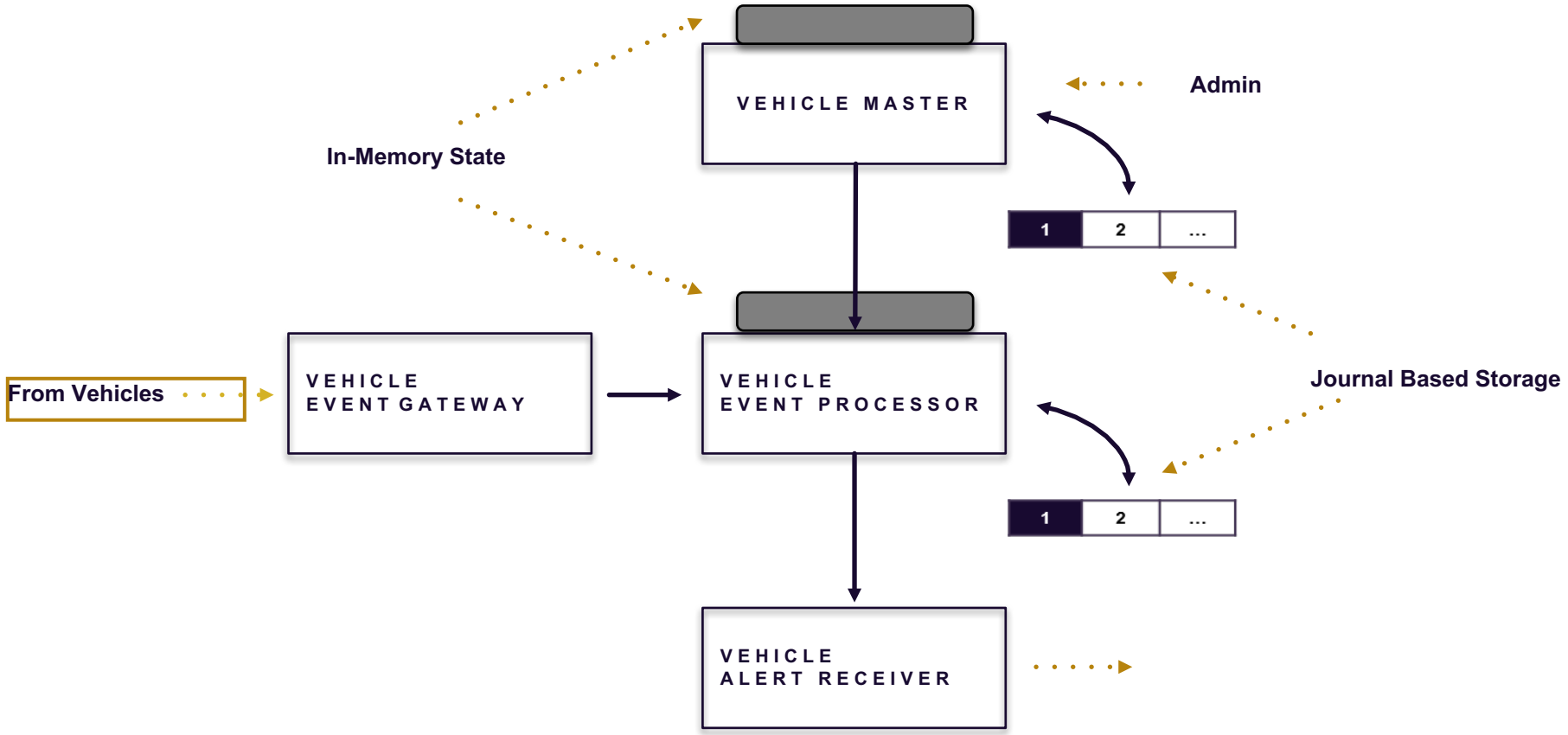
with

The X Platform

IMPLEMENTING GEOFENCING

- **We have a fleet of vehicles.**
  - (cars, trucks, whatever)
- **Each vehicle Should be following a route defined by Administrators**
- **Our Fleet Management System needs to:**
  - **Track location** of vehicles to ensure routes are being followed.
  - Monitor telemetry like speed, etc.
  - If a vehicle leaves its route, **trigger alerts**.

FLEET GEOFENCING



THE CODE

Message
Plain Old Java Object
Generated from XML Model

State Management
Plain Old Java Objects
Generated from XML Model

Messaging
Annotation based handler discovery,
Single Threaded

```
@EventHandler
final public void onLocationEvent(final LocationEventMessage message, final Repository state) throws Exception {
    try {
        validate(message);
    }
    catch (Exception e) {
        invalidEventCount.increment();
        return;
    }

    // look up vehicle's route:
    final VehicleRoute route = _vehicleRoutes.get(message.getVehicleID());
    if (route == null) {
        droppedCount.increment();
        return;
    }

    // update state
    Vehicle vehicle = state.getVehicles().get(message.getVehicleID());
    if (vehicle == null) {
        vehicle = Vehicle.create();
        vehicle.setVehicleID(message.getVehicleID());
        state.getVehicles().put(message.getVehicleID(), vehicle);
    }

    // check if vehicle is within its route bounds, and alert if not
    if (!updateVehicleLocationIfInRouteBoundary(vehicle, route, message.getLocation(), message.getSpeed())) {
        RouteViolationEvent alert = populateRoutViolationEvent(vehicle, message.getLocation(), message.getSpeed());
        _messageSender.sendMessage("alerts", alert);
        alertCount.increment();
    }

    // update stats
    processedCount.increment();
}
```

State Management
Plain Old Java objects and Java Collections

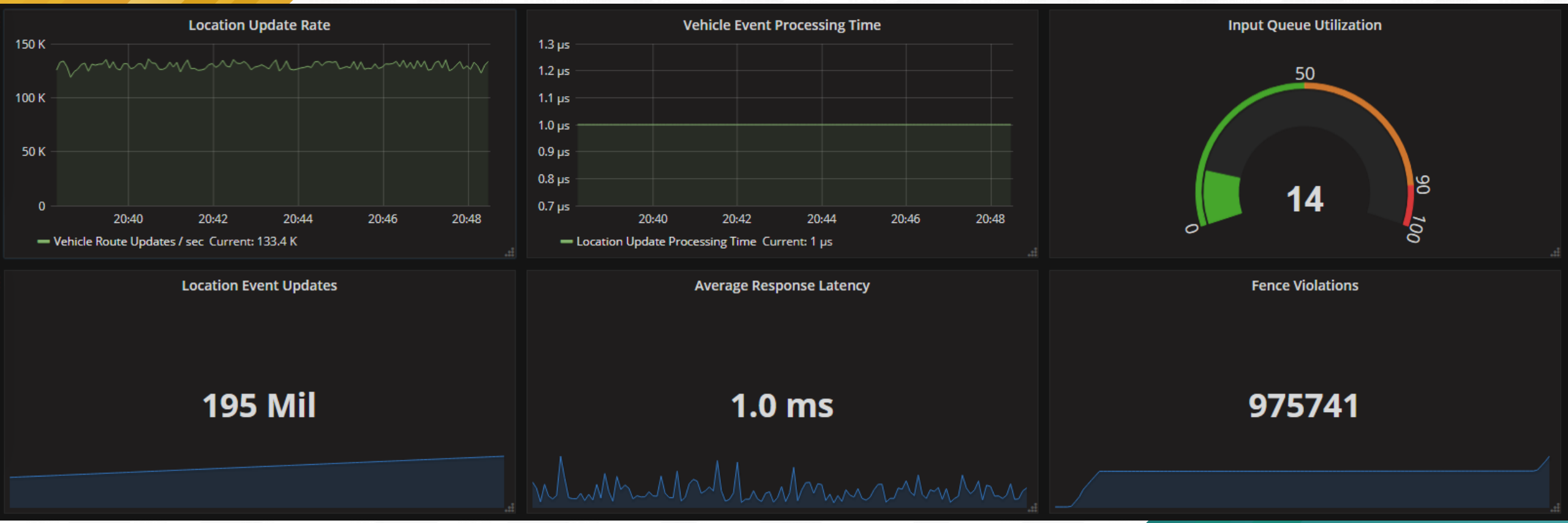
State Management
Object Pooling and Preallocation for Zero Garbage

State Management
State Changes transparently Replicated to Hot Backup and/or Disk Based Journal

Messaging
Create and populate "Fire and Forget"

Pure Business Logic – Exactly Once Processing

IOT FLEET GEOFENCING



Location Updates Events/sec: >130k

Single Shard, 1 Processor Core, Replicated.

1ms Response Time.

Full HA (Replicated), Exactly Once

In-Memory Computing SUMMIT EUROPE 2018

WHY X?

- **Easy to Build**
  - Focus on domain
    - Pure Java
- **Easy to Maintain**
  - Pristine domain
    - No infrastructure bleed
- **Easy to Support**
  - Stock hardware
  - Small Footprint
  - Simple abstractions
  - Easy tools
- **Very, very fast**

✓ No Compromise

Agility, Availability, Scalability, Performance

GETTING STARTED WITH X PLATFORM™

Getting Started Guide

https://docs.neeveresearch.com

Get the Demo Source

https://github.com/neeveresearch/nvx-apps

We're Listening

contact@neeveresearch.com

QUESTIONS