In-Memory Computing SUMMIT | EUROPE 2018

REAL-TIME WITH AI – THE CONVERGENCE OF BIG DATA AND AI

NEEVE RESEARCH
IN-MEMORY COMPUTING

COLIN MACNAUGHTON
NEEVE RESEARCH

INTRODUCTIONS

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

AGENDA

- MACHINE LEARNING: BIG DATA AND BETTER FEATURES
- PRODUCTIONIZING BIG DATA IN REAL TIME
- USE CASE: BIG DATA AND REAL WITH THE X PLATFORM

BIG DATA AND MACHINE LEARNING

Big Data and Machine Learning go Hand in Hand

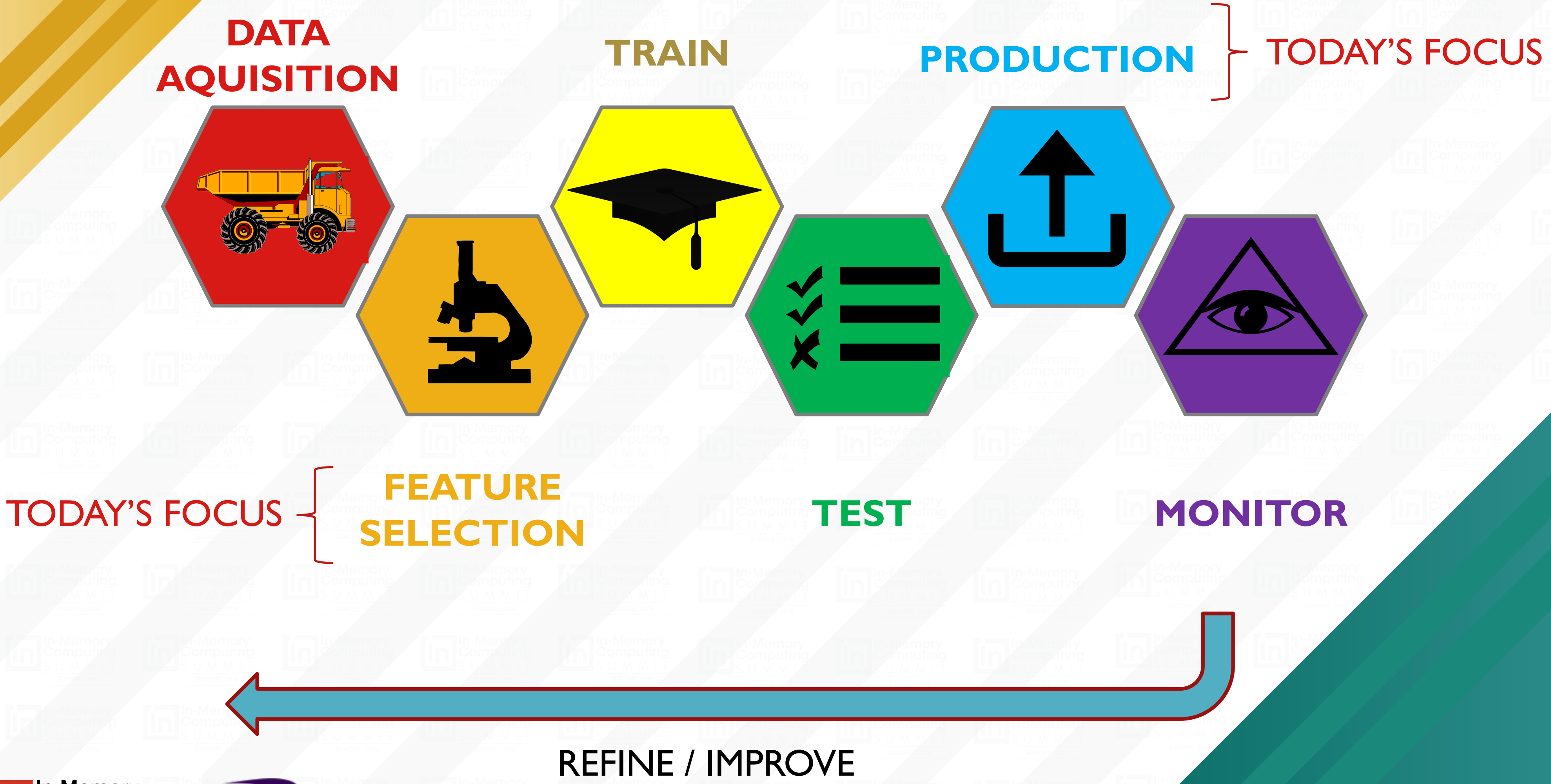
Training

- *Deep Learning has risen to the fore recently, and it is data hungry! When looking to make accurate predictions we need large data sets to train and test our models.*

In Production (real-time)

- *The more data (features) we can access and aggregate in real time to feed as inputs to our models, the more accurate our predictive output will be.*
- *This is an HTAP problem: can we assemble this data at scale while it is also being updated?*
- *Because models need to evolve continuously, loosely coupled (micro service) architectures are a good choice, but it means we'll be moving a lot of data around.*

MACHINE LEARNING WORKFLOW



In-Memory Computing SUMMIT EUROPE 2018

NEEVE RESEARCH IN-MEMORY COMPUTING

FEATURE SELECTION

It’s all about the data …but what data?

- Which pieces of data serve as the best predictors of what we are looking to answer?
- Can I get an accurate (enough) result just from the data in the request a user sent?
- If not can more data help?

FEATURE SELECTION

BIG DATA AND BETTER FEATURES

Can Big Data in Real Time help us leverage more meaningful features?

- *How much better are our predictive models if they can leverage features based on relevant historical/topical data on a transaction by transaction basis?*
- *Can we assemble such data within a meaningful time frame in production?*
- *Can we concurrently collect more data that we expect will be useful?*

FEATURE SELECTION

BIG DATA AND BETTER FEATURES

Example – Credit Card Fraud Detection

Feature	Big Data Enhanced Feature
Amount	Skew from median purchase, Amount charged in last hour.
Merchant	# of Prior Purchases by user
Location	Distance from last purchase? Distance from home(s)? Purchased from this location in the past?
Time	Last Purchase Time?

FEATURE SELECTION

BIG DATA AND BETTER FEATURES

Example – Personalization

Feature	Big Data Enhanced Feature
Time	Seasonal Interests / Habits … every year Jane goes snowshoeing in March.
Search Terms / Key words	Past Interests / Behavior
Location	• The last time John was in Paris, he was interested in… • John's calendar says he'll be in Paris next September. • X is happening here now (or in the future).
Demographics	What are peers clicking on now?

FEATURE SELECTION

MACHINE LEARNING IN PRODUCTION

Performance and Scale – Lots of data needed in real time

- Can I assemble the normalized feature data needed to feed my model in real time?
- Can I produce results fast enough that the prediction still matters?

Agility – Rapid Change: Models must evolve over time and so must the system feeding data to it.

- Fail Fast – Ability to rapidly test and discard what doesn't work.
- A/B testing
- Zero down time deployment, easy deployment to test environments.

High Availability

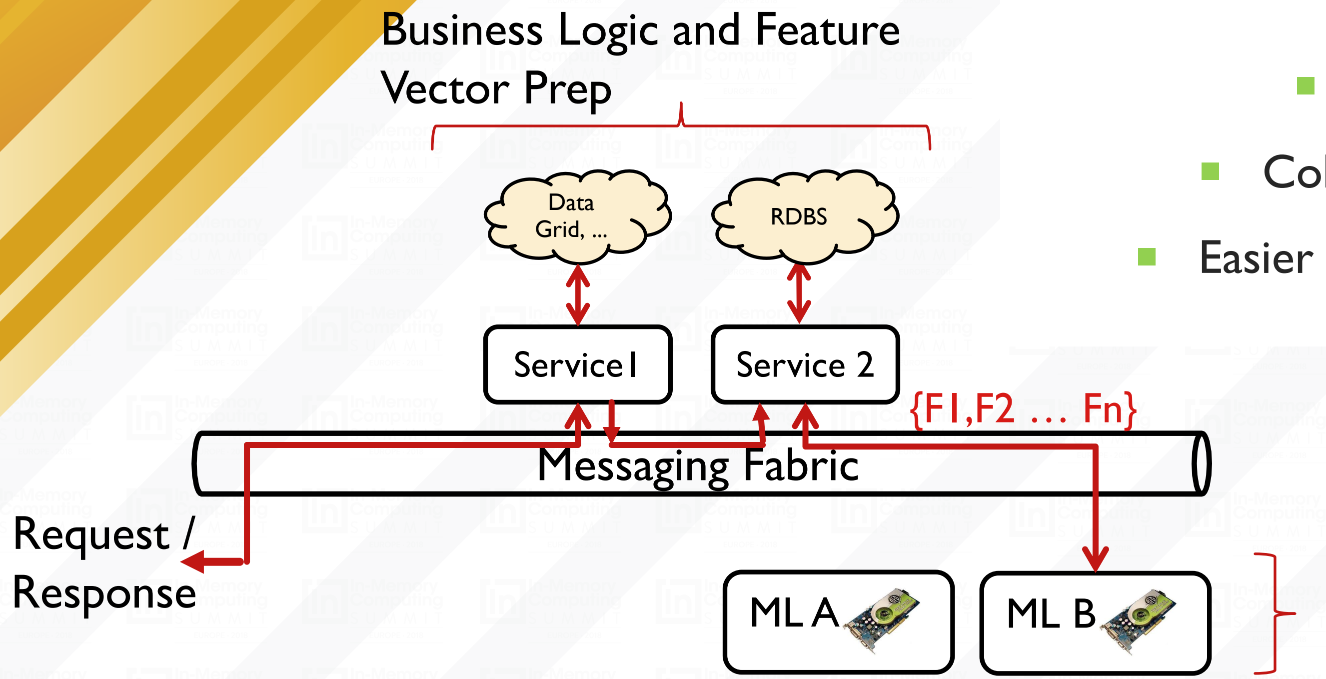
- No interruptions across Process, Machine or Data Center failure.

Business Logic

- ML isn't the answer to every problem, can your infrastructure handle traditional analytics and ML?
- Cyber Threats – Spooking the algorithm.

PRODUCTION

PLAN FOR (EVOLVING) SCALE – MICRO SERVICES



Micro Services:

- Each Service owns private state.
- Collaborate asynchronously via messaging
- Easier to scale + less contention on shared state

ML As Service
A/B testing made simple
w/ routing rules

Benefits

- Reduce Risk -> Increased Agility
- Cost Effective -> Provision to hardware by granular service needs.
- Resiliency -> Single service failure doesn't bring down the entire system.

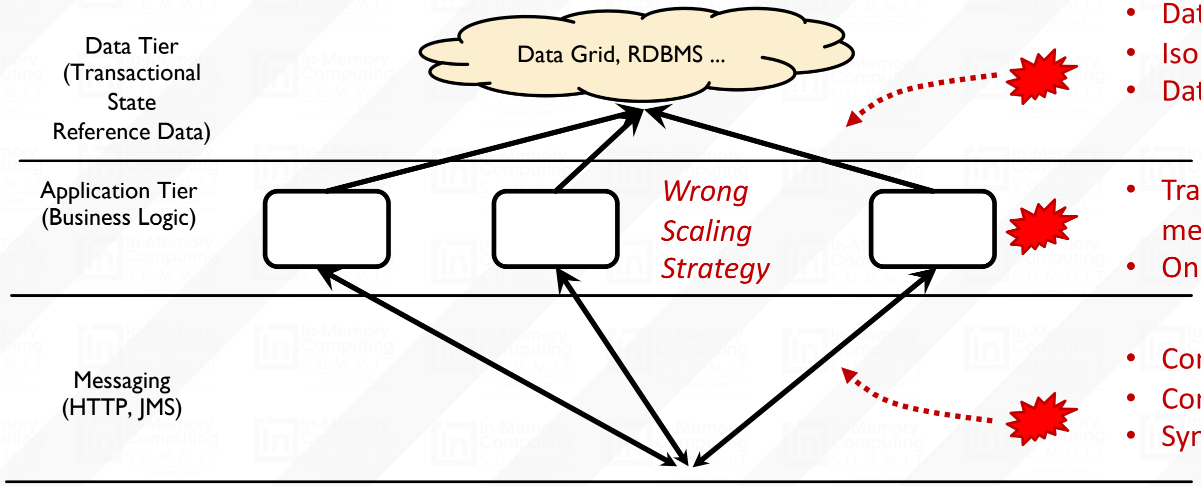
PRODUCTION

PLAN FOR (EVOLVING) SCALE – HA + DATA

Shared storage for HA and reliability

Launch more instances for scale + HA

Request Load Balancing



- Data Update Contention
- Isolation and Ordering
- Data Access Latency

- Transaction coordination between message and data stream.
- Only scales to a point.

- Complex Routing
- Complex Ordering
- Synchronous

Can you assemble the feature vectors needed to feed your model at scale?

- Not with the above … Update Contention betweens threads / instances prevents the ability to do big data reads.

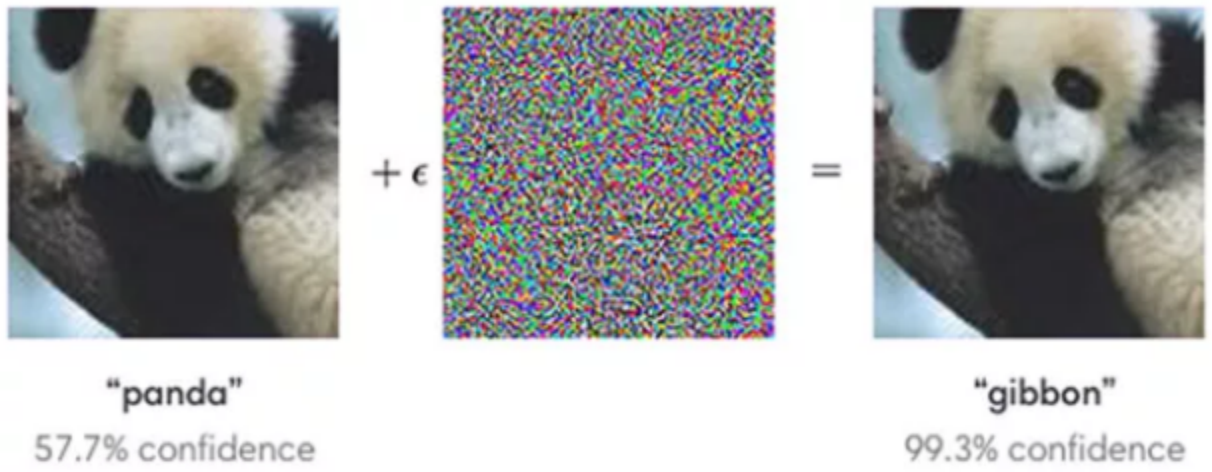
PRODUCTION

DON'T FORGET PLAIN OLD BUSINESS LOGIC

Traditional Analytics are Still Important!

- Not all analytics are best solved with ML … be judicious.
- Deep Neural Networks are a Black Box…
- … so when possible traditional rules/analytics should complement ML, along with robust monitoring.

Example: Adversarial Inputs



An unmodified image of panda (left), when mixed with a finely tuned "perburbation" (center), makes AIs think it's a gibbon (right).
Image: OpenAI/Google Brain

PRODUCTION

PLAN WORKFLOW FOR REFINEMENT

- Plan for measuring and monitoring ML efficacy
  - Behavior changes over time
  - Models will need to evolve.
- Getting data out
  - Consider infrastructural / security implications of exposing production data for refinement training of models.
  - Continuous training workflows?

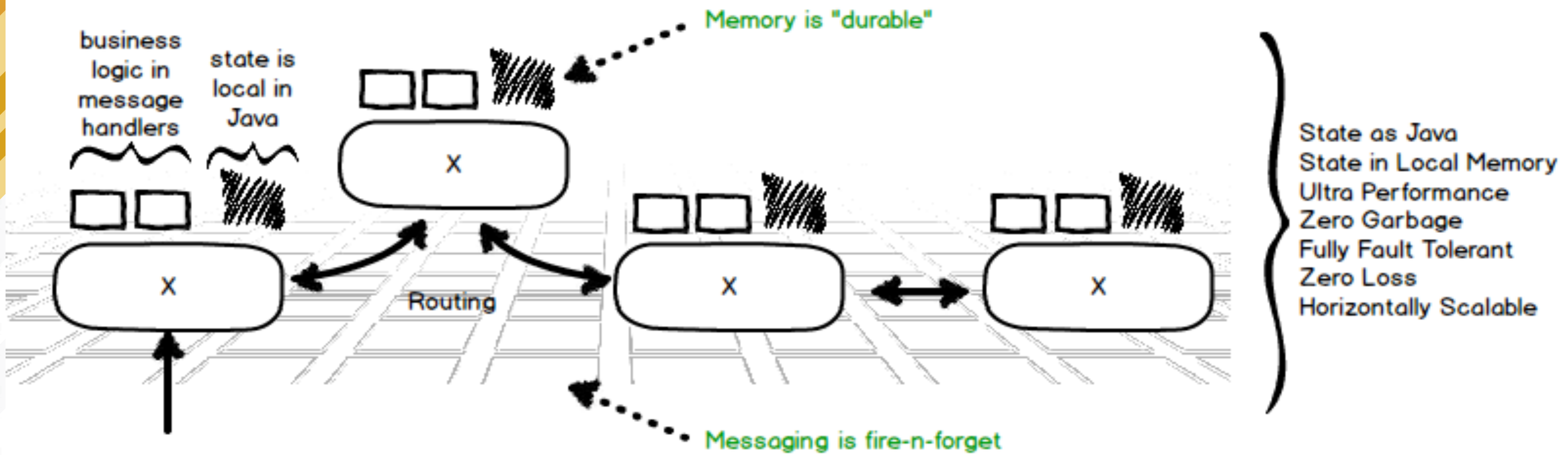
DATA AQUISITION

THE X PLATFORM

THE X PLATFORM

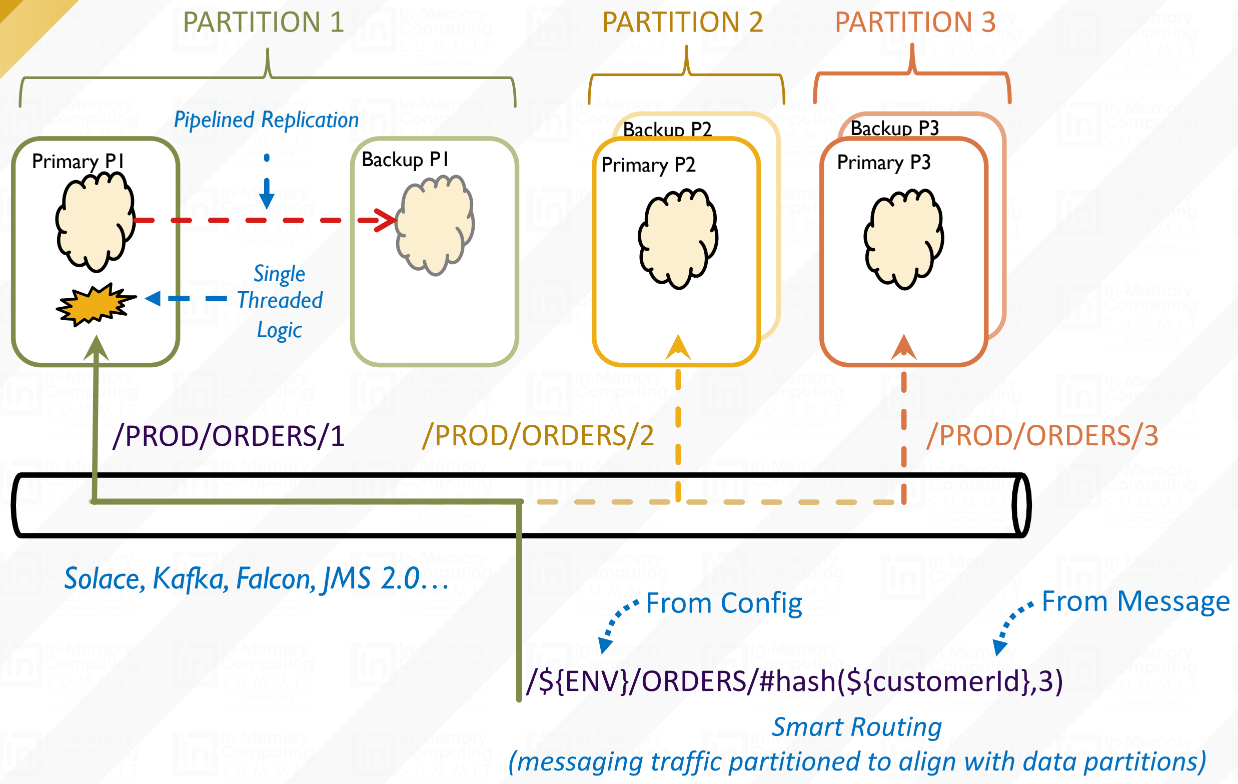
The X Platform is a memory oriented platform for building *multi-agent, transactional* applications.

Collocated Data + Business Logic = Full Promise of In-Memory Computing



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

TRANSACTION PROCESSING WITH X PLATFORM



KEY TAKEAWAYS

DATA:

- **STRIPED** – NO UPDATE CONTENTION, HORIZONTAL SCALE
- **IN MEMORY** – NO DATA ACCESS LATENCY, DISK BASED JOURNAL BACKED
- **PLAIN OLD JAVA OBJECTS**– FLEXIBLE, EVOLVABLE ENCODING

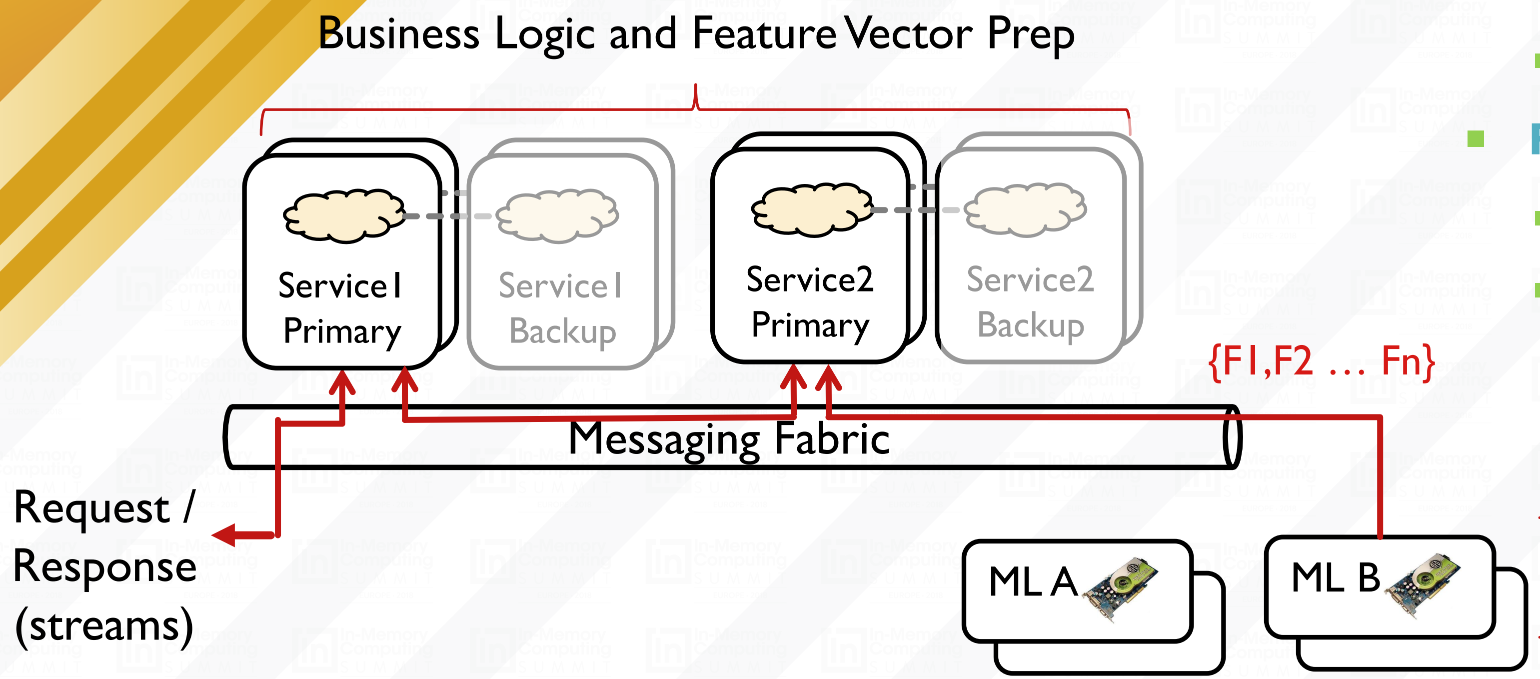
MESSAGING

- **CONTENT BASED** – TRANSPARENT ROUTING TO DATA
- **FIRE AND FORGET** – EXACTLY ONCE PROCESSING, CONSISTENT WITH STATE
- **PLAIN OLD JAVA OBJECTS**– FLEXIBLE, EVOLVABLE ENCODING

HIGH AVAILABILITY

- **PIPELINED REPLICATION** – NON BLOCKING PIPELINED MEMORY-TO-MEMORY -> STREAM TRANSACTION PROCESSING
- **NO DATA LOSS** – ACROSS PROCESS, MACHINE, DATA CENTER FAILURE

WHAT DOES THIS MEAN FOR ML + BIG DATA IN REAL TIME?

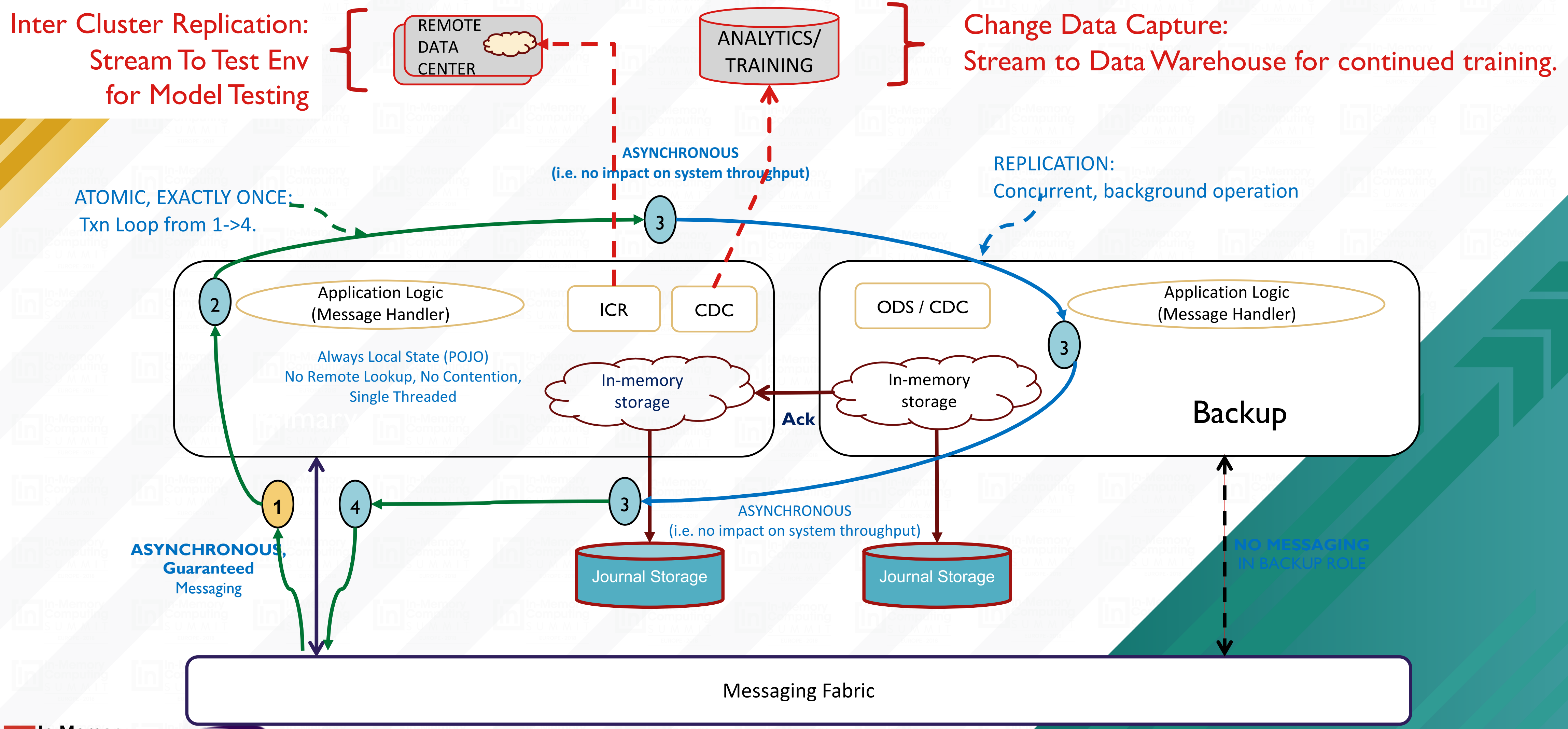


ML As Service
A/B testing made simple
w/ routing rules

- **SCALABLE**
  - **By Partitioning**
- **FAST!**
  - **All Data In Memory (no remoting)**
  - **No Data Contention (Single Thread)**

- **AGILITY**
  - **Micro Service Architecture**
  - **Trivial evolution of message + data models**
- **HA**
  - **Memory-Memory Replication Pipelined, Async Journal Backed.**
  - **Exactly Once Delivery across failures**

DATA WORKFLOWS



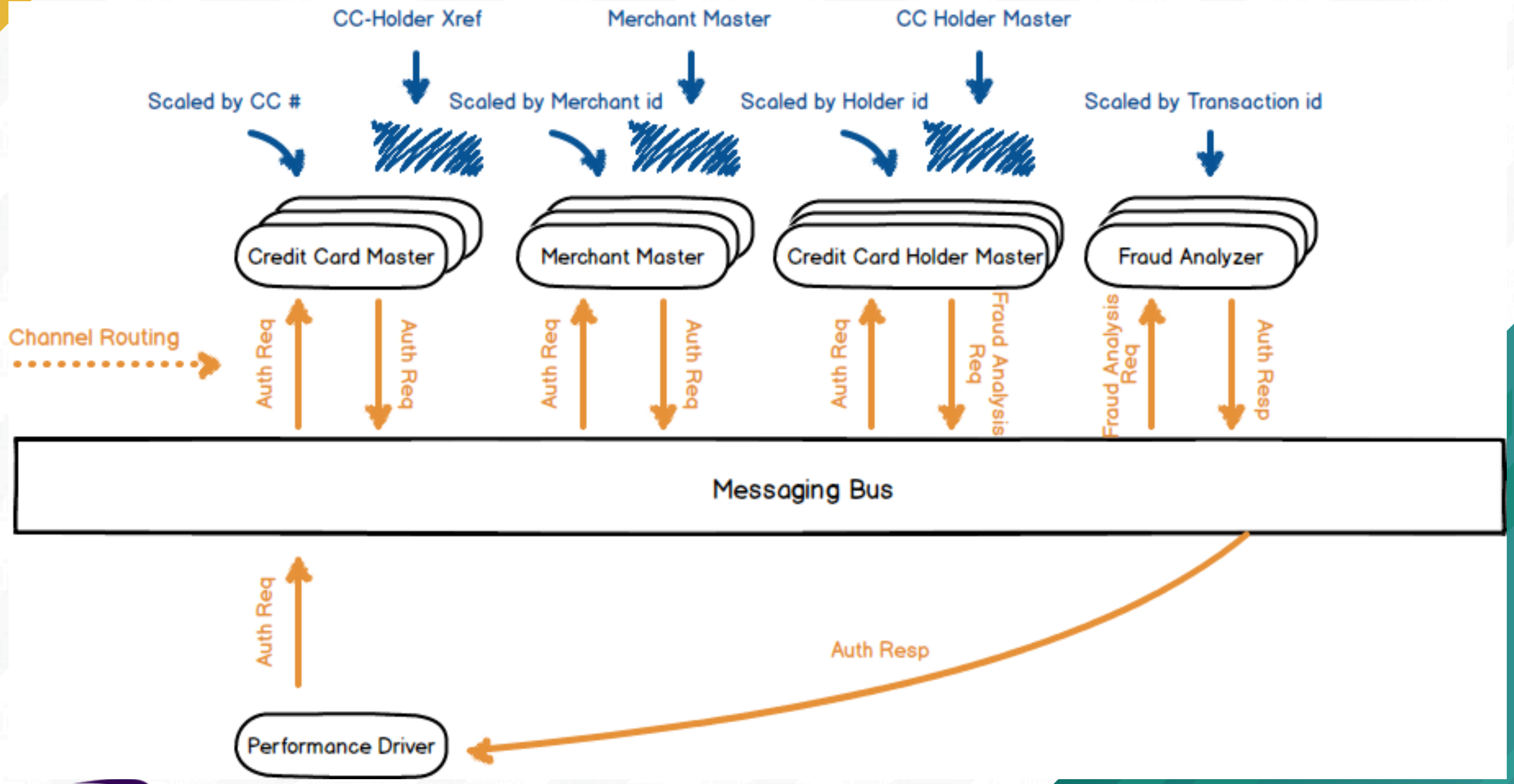
USE CASE - REAL TIME FRAUD DETECTION

- Receive CC Authorization Request
  - Identify Card Holder
  - Identify Merchant
  - Perform Fraud Checks using
    - CC Holder Specific Information
    - Transaction History
- Send CC Authorization Response

Reference Data Aggregation

Hybrid Rule Based Analytics + Machine Learning

FLOW



CC-Holder Xref
Merchant Master
CC Holder Master
Scaled by CC #
Scaled by Merchant id
Scaled by Holder id
Scaled by Transaction id
Credit Card Master
Merchant Master
Credit Card Holder Master
Fraud Analyzer
Channel Routing
Auth Req
Auth Req
Auth Req
Auth Req
Auth Req
Fraud Analysis Req
Fraud Analysis Req
Auth Resp
Messaging Bus
Auth Req
Auth Resp
Performance Driver

PERFORMANCE

200k Merchants

100k Credit Cards

35 million Transactions

TensorFlow (no GPU)

2 Partitions, Full HA

7500k auth/sec

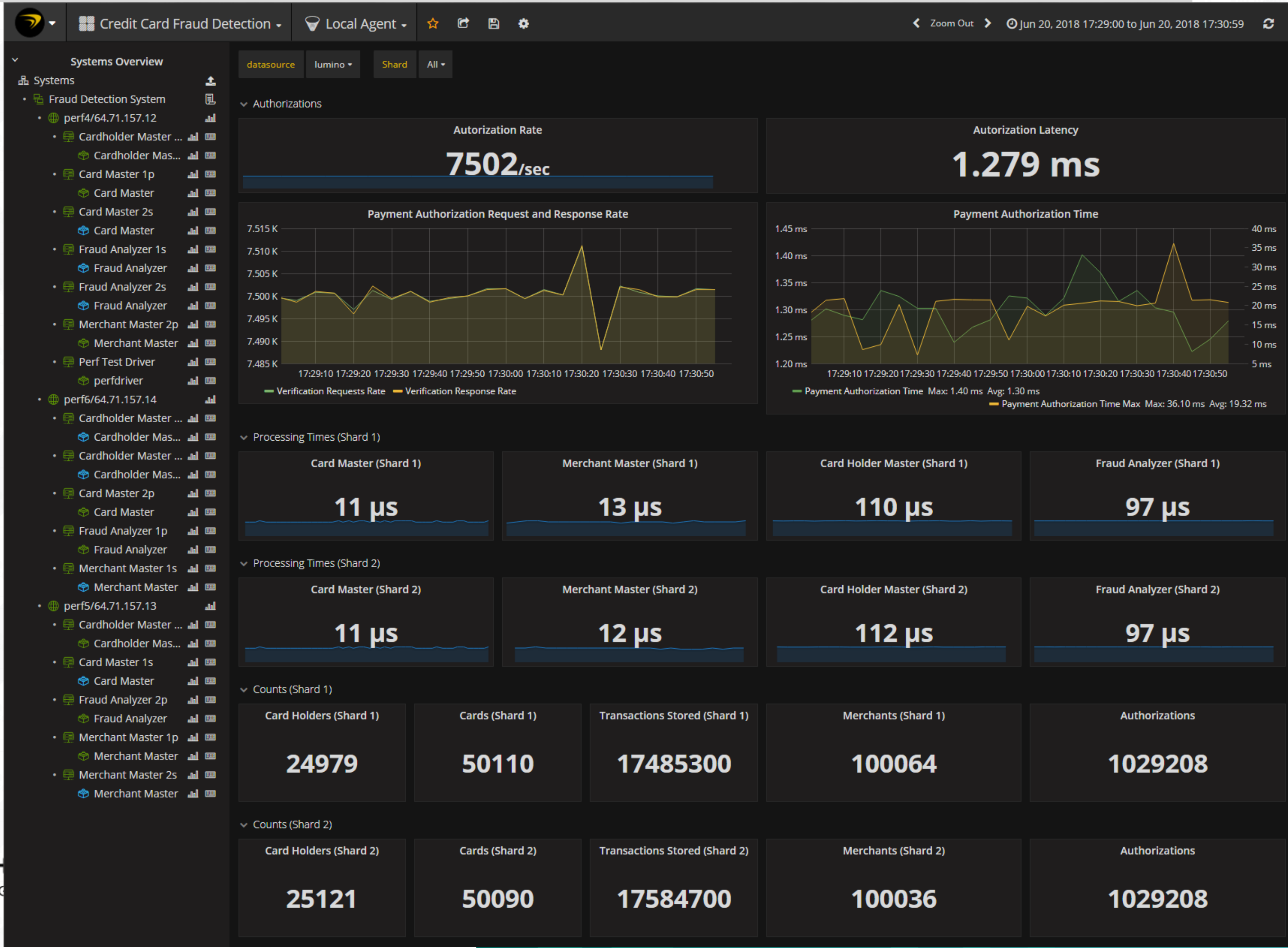
Auth Response Time = ~1.2ms

FRAUD DETECTION WITH TENSOR FLOW

50k Credit Cards / Instance
17.5m Transactions / Shard
100k Merchants / Shard

1.2ms median Authorization Time (36.4 ms max)

Full Scan of one year's worth of transactions per card on each authorization to feed ML



In-Memory Computing SUMMIT EUROPE 2018

NEEVE RESEARCH IN-MEMORY COMPUTING

HAVE A LOOK FOR YOURSELF

Check Out the Source

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

Getting Started Guide

https://docs.neeveresearch.com

Get in Touch

contact@neeveresearch.com

QUESTIONS