Real-Time with AI
The Convergence of Big Data and AI

Colin MacNaughton
Neeve Research
INTRODUCTIONS

• Based here 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 -> BETTER FEATURES

• PRODUCTIONIZING BIG DATA IN REAL TIME

• USE CASE: REAL TIME FRAUD DETECTION
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/HOAP 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 at the risk of needing to move a lot of data around.
TYPES OF APPLICATIONS

- Financial Trading
- IoT Event Processors
- Credit Card Processors
- E-Commerce
  - Personalization Engines
  - Value Based Pricing
- Ad Exchanges
- ...

Neeve Research
MACHINE LEARNING WORKFLOW

DATA ACQUISITION

TRAIN

FEATURE SELECTION

TEST

PRODUCTION

MONITOR

REFINE / IMPROVE

TODAY’S FOCUS
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?
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?
### Example – Credit Card Fraud Detection

<table>
<thead>
<tr>
<th>Feature</th>
<th>Big Data Enhanced Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>Skew from median purchase, Amount charged in last hour.</td>
</tr>
<tr>
<td>Merchant</td>
<td># of Prior Purchases by user</td>
</tr>
<tr>
<td>Location</td>
<td>Distance from last purchase? Distance from home(s)? Purchased from this location in the past?</td>
</tr>
<tr>
<td>Time</td>
<td>Last Purchase Time?</td>
</tr>
</tbody>
</table>
# BIG DATA AND BETTER FEATURES

## Example – Personalization

<table>
<thead>
<tr>
<th>Feature</th>
<th>Big Data Enhanced Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Seasonal Interests / Habits ... every year Jane goes snowshoeing in March.</td>
</tr>
<tr>
<td>Search Terms / Key words</td>
<td>Past Interests / Behavior</td>
</tr>
</tbody>
</table>
| Location                 | • The last time John was in Paris, he was interested in...  
                          | • John’s calendar says he’ll be in Paris next September.  
                          | • XYZ is happening here now (or in the future).          |
| Demographics             | What are peers clicking on now? |
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 compute/data infrastructure handle traditional analytics and ML?
• Cyber Threats – duping the model.
Can you assemble the feature vectors needed to feed your model at scale?

- Not with the above ... Update Contention between threads / instances prevents the ability to do big data reads.
PLAN FOR (Evolving) SCALE – COMPUTE + Data + HA

Data Tier
(Transactional State Reference Data)

Application Tier
(Business Logic)
Collapsed

Messaging
(Publish -Subscribe)

Processing Swim-lanes (ordered)

Routing Strategy?

In-Memory + Partitioned

+ Co-located Function + Data

+ Replicated

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
Pick up feature data in streaming processing pipeline.

Business Logic and Feature Vector Prep

{F1,F2 ... Fn}

Request / Response

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.
Data to aggregate across lots of disparate Microservices?

Parallel Fetch (Fork/Join)
- choice of messaging provider matters, but modern providers can handle it.

Request / Response

ML A  ML B

{F1, F2 ... Fn}
What Happens when Services are Updated?

- Choice of message encoding is critical.
  - Older versions of services should still function when new fields added.
- Efficiency of Encoding Matters!
- Impedance mismatch between State/Message encoding?
- Organization-wide agreed upon “Rules of Engagement”
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 “perturbation” (center), makes AI think it’s a gibbon (right).

Image: OpenAI/Google Brain
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?
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
HA + SCALE ON THE X PLATFORM

PARTITION 1
Primary P1
Backup P1

PARTITION 2
Backup P2
Primary P2

PARTITION 3
Backup P3
Primary P3

Solace, Kafka, Falcon, JMS 2.0...

/PROWD/ORDERS/1
/PROWD/ORDERS/2
/PROWD/ORDERS/3

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

From Config
/$(ENV)/ORDERS/#hash($(customerId),3)

From Message
Smart Routing
(messaging traffic partitioned to align with data partitions)
WHAT DOES THIS MEAN FOR ML + BIG DATA IN REAL TIME?

SCALABLE
- By Service 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 (Zero Down Time)
- Exactly Once Delivery across failures (Zero Duplication/Loss)
Getting Data Out…

Change Data Capture:
Stream to Data Warehouse for continued training.

Inter Cluster Replication:
Stream To Test Env for Model Testing

Application Logic (Message Handler)
Always Local State (POJO)
No Remote Lookup, No Contention, Single Threaded

In-memory storage

Journal Storage

Messaging Fabric

REMOTE DATA CENTER

APPLICATION/ TRAINING

ASYNCHRONOUS
(i.e. no impact on system throughput)

REPLICATION:
Concurrent, background operation

ASYNCHRONOUS, Guaranteed Messaging

NO MESSAGING IN BACKUP ROLE
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
FRAUD DETECTION WITH THE X PLATFORM + 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 two year’s worth of transactions per card on each authorization to feed ML

Neeve Research
Performance Summary for 2 Partitions

- 200k Merchants
- 100k Credit Cards
- 35 million Transactions
- TensorFlow (no GPU)
- 2 Partitions, Full HA
- 7500k auth/sec

Auth Response Time = ~1.2ms
HAVE A LOOK FOR YOURSELF

Check Out the Source
https://github.com/neeverereasearch/nvx-apps

Getting Started Guide
https://docs.neeverereasearch.com

Get in Touch
contact@neeverereasearch.com
Questions