Policing The Capital Markets with ML

Cliff Click
CTO Neurensic
cclick@neurensic.com

Who Am I?

Cliff Click

CTO Neurensic

Co-Founder H2O.ai

cliffc@acm.org

PhD Computer Science

1995 Rice University

HotSpot JVM Server Compiler

“showed the world JITing is possible”

45 yrs coding

40 yrs building compilers

35 yrs distributed computation

30 yrs OS, device drivers, HPC, HotSpot

15 yrs Low-latency GC, custom java hardware, NonBlockingHashMap

20 patents, dozens of papers

100s of public talks

neurensic™

Neurensic

Our Global Financial Markets Are In Trouble

100% of our economy — is valued and priced by the capital markets, which include stocks, futures, options and other financial instruments

84% of stock trading — on all US markets is done by sophisticated, high-frequency trading computer algorithms

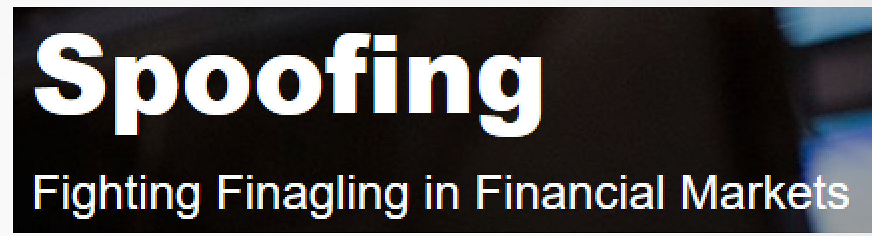
70% of futures trading — on CME foreign exchange, equities and interest rate markets is done by advanced automated trading systems

<1% of trade surveillance — solutions (excluding Neurensic) use machine learning or pattern recognition to understand automated trading activity

$900,000,000 dollars — is spent annually by financial institutions on trade surveillance alone, with tens of billions on compliance overall

Neurensic – Forensics in the Markets

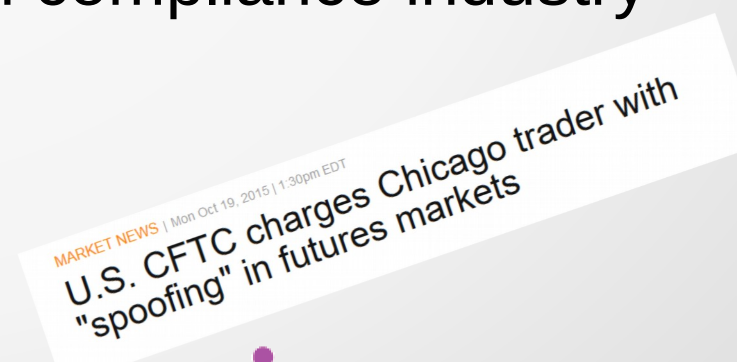
- Neurensic specializes in Market Forensics
- Reads Financial Data Streams aka stock "ticker tape"
- Looks for Illegal Activity
- Tooling, not law enforcement
  - Tool is used by regulators, mutual funds, FCMs, traders
- Addresses a $Tn problem in a $Bn compliance industry



The Situation

In July, the first person to be convicted of spoofing, high-speed commodities trader Michael Coscia, was sentenced to three years in prison by a U.S. judge in Chicago. It took a jury only an hour to convict him in November 2015. A British futures trader, Navinder Sarao, is facing extradition to the U.S. after he was arrested in suburban London in April 2015. U.S. authorities said Sarao's activities contributed to the flash crash of May 2010, when almost $1 trillion was temporarily wiped out in the U.S. stock market. While reports of spoofing

$1,000,000,000,000



neurensic™

Financial Data: The “ticker tape”

- Not just NYSE Ticker Tape
  - “Tickers” from CME and all exchanges
  - Audit logs, clearing houses, internal trading systems
- Financial Data is Big Data:
  - World-wide probably 1Trillion rows daily for Futures
  - Big firm might see 1Billion rows daily
    - About 1Tbyte daily
  - Common to see 10m rows, 10Gig daily
- Need to run sophisticated ML algorithms
- Algos change rapidly to follow the crooks - “arms race”
- Lots of unusual 1-off feature generation

neurensic™

Results as Risk

- Dodd-Frank - “Intent to Deceive” is illegal
- Neurensic builds tools; does not declare “intent”
  - (that requires a judge)
- Results couched as “Risk”:
  - Risk == odds of behavior considered illegal
  - Basically: activities in the market similar to what has been investigated or prosecuted already
- Machine Learning: find close matches to patterns in data
- Investigation by a Compliance Officer next

200
“Safe”

800
“Risky”

Requirement to be Transparent

- Computers do not declare “guilty”, legal system does
- All parties need to understand the data
- Finding an questionable activity is just the first step!
- Now need to explain **why** it's questionable
- Machine Learning notorious for being opaque (but correct)
- How do we justify ML results to a Federal Judge?
- Answer: **we don't**.
- We find interesting patterns *<u>and show them</u>*

neurensic™

Explaining Market Data

- We show what the trading firm knows
  - Internal Audit Logs
    - Trader activity over time, attempts to trade
    - “Position” - accumulations of stocks/futures
    - Buy/Sell offers
- We show what the public market knows:
  - “Ticker” data; bid/ask spread; volume traded
  - Canceled offers, historical trends
- And we must filter, **filter**, ***filter*** down to human scale
  - Billions must become 100's of rows

neurensic™

Visualization of Raw Data is Key

- Must use the actual ticker/audit data, not ML results
  - Because this is understood, and hard legal evidence
  - Data is messy, “symbology” changes over time, place
  - Data is too big to look at; needs to be filtered, reduced
- Must visualize the patterns:
  - Show trades in real time, slow time, tick-by-tick time
  - Matching trader positions, activities, bids/offers/cancels
  - “The Book” - outstanding market bids/asks
  - Visual displays of all of the above, over time
    - “Movies” of abstract financial trades

neurensic™

Rapid Evolution of Displays

- We need to improve existing displays
  - Better visuals for existing suspicious patterns
  - Better filtering (always a tension between too little and too much)
  - Legal requirements change
- We need to add new displays
  - New visuals for new patterns
    - As old patterns get stopped, new ones emerge
- Displays moving from rich desktop to browser to mobile

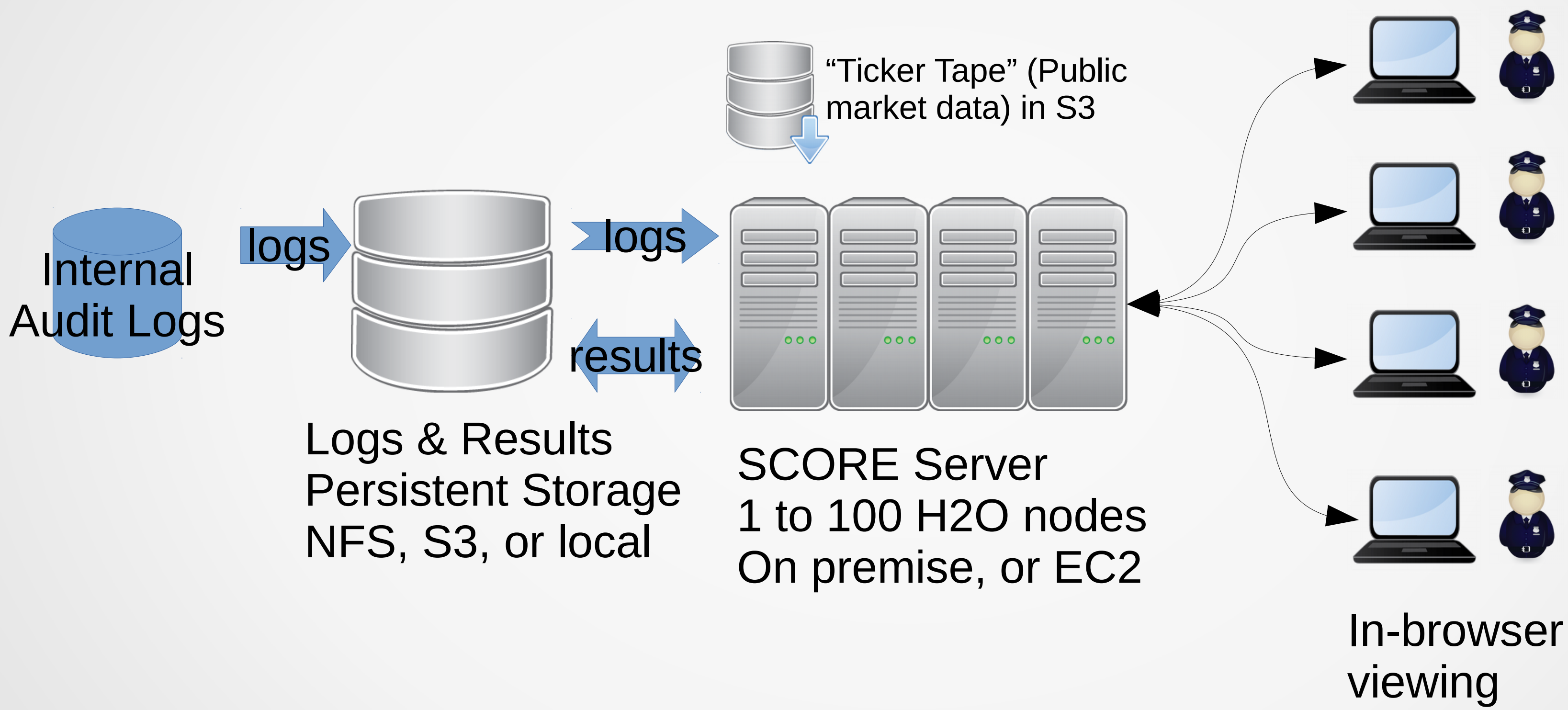
neurensic™

Modernize Displays

- Moving from thick-client desktop to browser
  - Browsers are everywhere
  - No install needed of thick-client
  - Bring html safely through firewalls (VPN)
- Allow mobile clients in the future
  - Show results to CxO's or lawyers
  - Quick check of own trading behavior
- And split server from client
  - Data inside corp private datacenter; Server with data
  - Client is many places

neurensic™

SCORE Architecture



H2O and Machine Learning

- H2O.ai is a premier open source ML tool
- Datasizes involved are easily within H2O's size
  - 10G to 40G on a single server
  - Terabyte on a modest cluster
- ML algorithms are bleeding-edge start of the art
- Direct implementations for Python and R
- All Neurensic's Data Science is done with Python
  - Taking DS algos direct from research to production

neurensic™

SCORE Internal Design

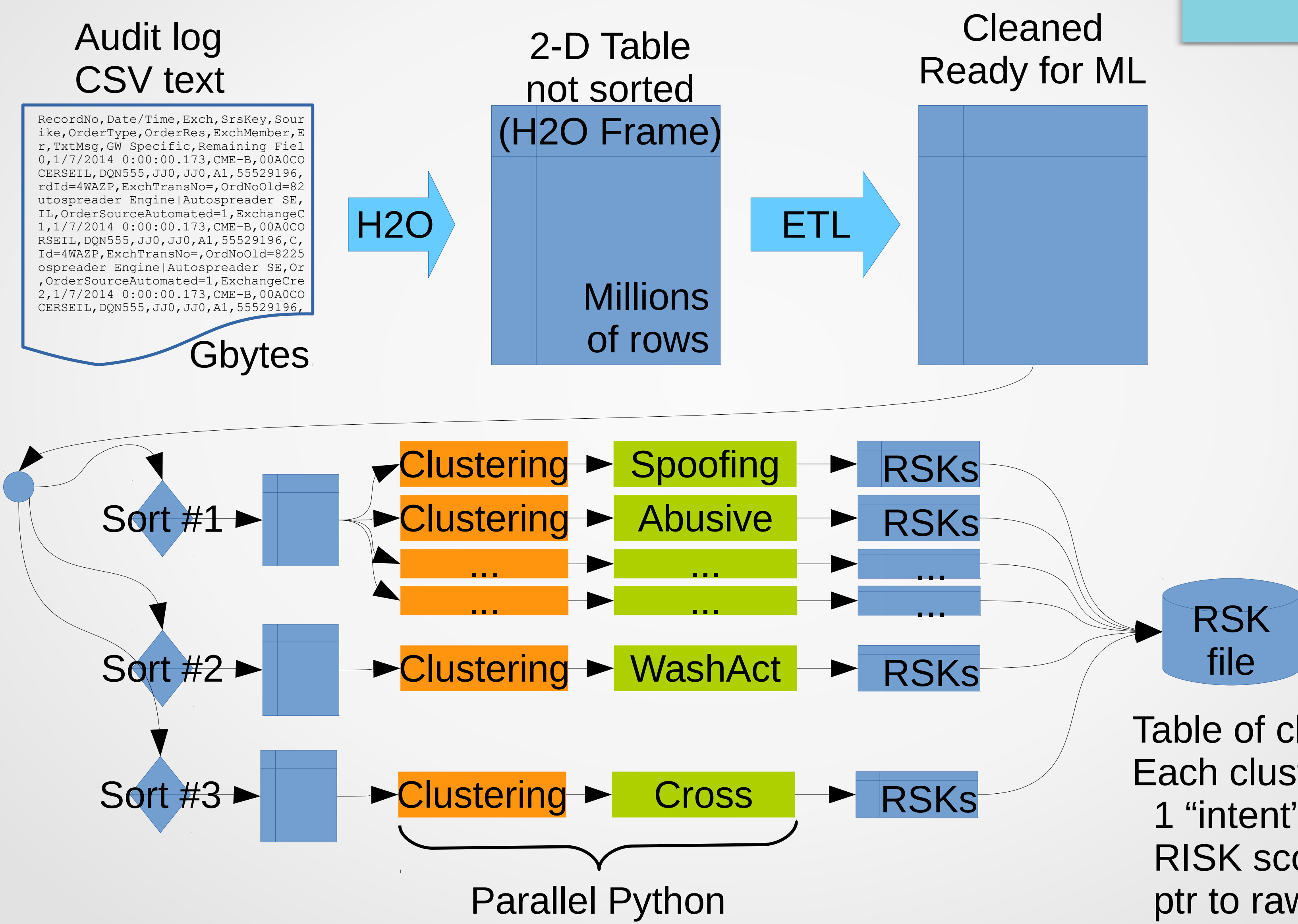
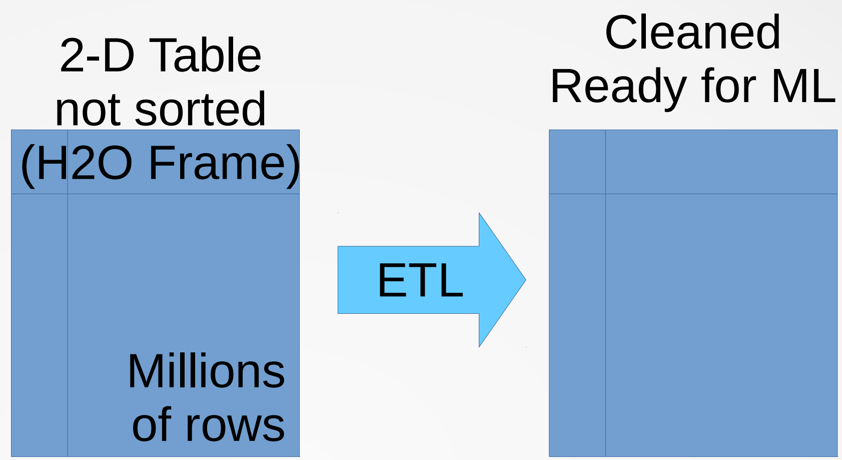


Table of clusters
Each cluster is:
1 “intent”
RISK score
ptr to raw data
ML vectors

ETL – Data Cleaning

- Read audit log
- Decide Vendor
  - TT, CQG, CME Audit, …

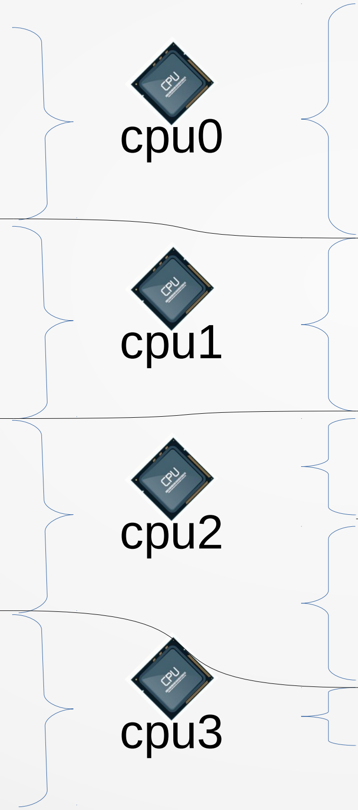


- Vendor specific ETL
  - Drop or impute missing values
  - Exchange, product, price normalization
  - Trader & account normalization
  - Uniform mapping for tokens
    - e.g. `{B,Buy,BUY}` → `Buy`; `{Limit,LMT,L,K,2}` → `Limit`
  - 100s of individual cleanup steps

Parallel Clustering – Python & Java

- Data ETL'd & cleaned; sorted already

Sym	Time	Action	Price
NDAQ	1:23.456	Add	78.9
NDAQ	1:23.457	Add	79.0
NDAQ	1:23.458	Add	78.7
NDAQ	1:23.459	Add	78.9
NDAQ	1:23.459	Fill	78.7
NDAQ	1:23.461	Reject	78.9
NDAQ	1:23.463	Cancel	78.9
NDAQ	1:23.463	Add	78.9
NDAQ	1:45.678	Fill	76.5
NDAQ	1:45.678	Add	76.5
NDAQ	1:45.679	Fill	78.9
NDAQ	1:45.680	Reject	78.9
NDAQ	1:45.680	Cancel	78.9
NDAQ	1:45.681	Add	78.9
NDAQ	1:55.681	Fill	78.9
NDAQ	1:55.681	Add	78.9
NDAQ	1:55.682	Add	78.9
NDAQ	1:55.683	Fill	78.9
AAPL	1:55.684	Reject	78.9
AAPL	1:55.684	Cancel	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Fill	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Add	78.9
AAPL	2:01.684	Add	78.9
AAPL	2:01.684	Add	78.9



Sym	Time	Action	Price
NDAQ	1:23.456	Add	78.9
NDAQ	1:23.457	Add	79.0
NDAQ	1:23.458	Add	78.7
NDAQ	1:23.459	Add	78.9
NDAQ	1:23.459	Fill	78.7
NDAQ	1:23.461	Reject	78.9
NDAQ	1:23.463	Cancel	78.9
NDAQ	1:23.463	Add	78.9
NDAQ	1:45.678	Fill	76.5
NDAQ	1:45.678	Add	76.5
NDAQ	1:45.679	Fill	78.9
NDAQ	1:45.680	Reject	78.9
NDAQ	1:45.680	Cancel	78.9
NDAQ	1:45.681	Add	78.9
NDAQ	1:55.681	Fill	78.9
NDAQ	1:55.681	Add	78.9
NDAQ	1:55.682	Add	78.9
NDAQ	1:55.683	Fill	78.9
AAPL	1:55.684	Reject	78.9
AAPL	1:55.684	Cancel	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Fill	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Add	78.9
AAPL	2:01.684	Add	78.9
AAPL	2:01.684	Add	78.9

- Each cpu does roughly equal work

Parallel Clustering – Python & Java

Sym	Time	Action	Price
NDAQ	1:23.456	Add	78.9
NDAQ	1:23.457	Add	79.0
NDAQ	1:23.458	Add	78.7
NDAQ	1:23.459	Add	78.9
NDAQ	1:23.459	Fill	78.7
NDAQ	1:23.461	Reject	78.9
NDAQ	1:23.463	Cancel	78.9
NDAQ	1:23.463	Add	78.9
NDAQ	1:45.678	Fill	76.5
NDAQ	1:45.678	Add	76.5
NDAQ	1:45.679	Fill	78.9
NDAQ	1:45.680	Reject	78.9
NDAQ	1:45.680	Cancel	78.9
NDAQ	1:45.681	Add	78.9
NDAQ	1:55.681	Fill	78.9
NDAQ	1:55.681	Add	78.9
NDAQ	1:55.682	Add	78.9
NDAQ	1:55.683	Fill	78.9
AAPL	1:55.684	Reject	78.9
AAPL	1:55.684	Cancel	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Fill	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Add	78.9
AAPL	2:01.684	Add	78.9
AAPL	2:01.684	Add	78.9

- Clustering rules in Python
  - Good for DS team!
- Python per row:
  - {keep,drop,start new cluster}
- Execution in parallel Jython
  - Fast on Big Data

Parallel Clustering – Python & Java

- CPU reads ~100k rows, builds ~1k clusters of ~100 rows each
- Clusters are: same instrument, close in time, but model-specific
- Represent ***intent***
- Clusters vary:
  - Wash Trade is 2 rows, Abusive Messaging might be 10000
  - Wash is 1msec; Spoof might be 5min

cpu0

cpu1

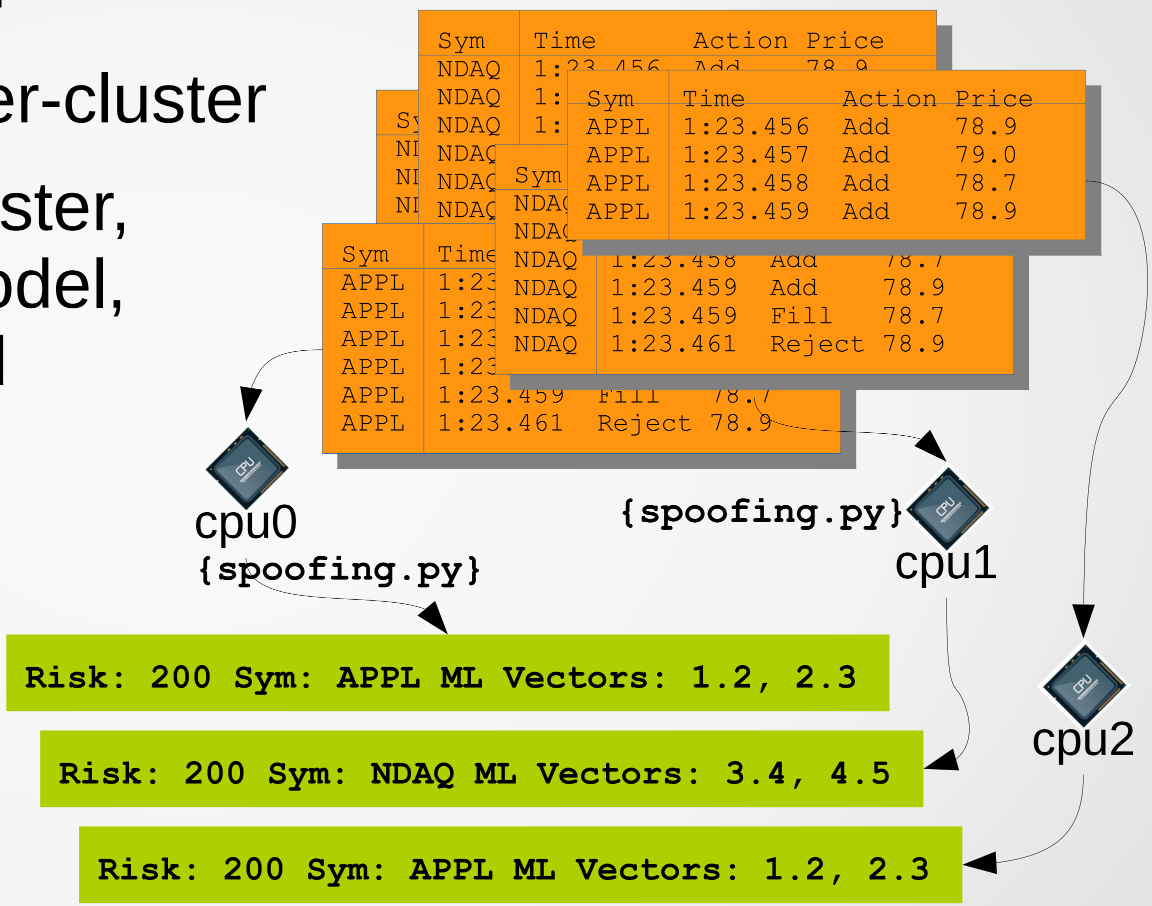
cpu2

cpu3

Sym	Time	Action	Price
NDAQ	1:23.456	Add	78.9
NDAQ	1:23.457	Add	79.0
NDAQ	1:23.458	Add	78.7
NDAQ	1:23.459	Add	78.9
NDAQ	1:23.459	Fill	78.7
NDAQ	1:23.461	Reject	78.9
NDAQ	1:23.463	Cancel	78.9
NDAQ	1:23.463	Add	78.9
NDAQ	1:45.678	Fill	76.5
NDAQ	1:45.678	Add	76.5
NDAQ	1:45.679	Fill	78.9
NDAQ	1:45.680	Reject	78.9
NDAQ	1:45.680	Cancel	78.9
NDAQ	1:45.681	Add	78.9
NDAQ	1:55.681	Fill	78.9
NDAQ	1:55.681	Add	78.9
NDAQ	1:55.682	Add	78.9
NDAQ	1:55.683	Fill	78.9
AAPL	1:55.684	Reject	78.9
AAPL	1:55.684	Cancel	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Fill	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Add	78.9
AAPL	2:01.684	Add	78.9
AAPL	2:01.684	Add	78.9

Parallel Python ML Modeling

- Clusters run in parallel
  - Run sequentially per-cluster
- Each CPU grabs a cluster, runs Python (Java) model, builds ML vectors, and scores for risk
- Work varies by model and cluster size
  - Worklist load balances



Parallel Python ML Modeling

Sym	Time	Action	Price
NDAQ	1:23.456	Add	78.9
NDAQ	1:23.457	Add	79.0
NDAQ	1:23.458	Add	78.7
NDAQ	1:23.459	Add	78.9
NDAQ	1:23.459	Fill	78.7
NDAQ	1:23.461	Reject	78.9
NDAQ	1:23.463	Cancel	78.9
NDAQ	1:23.463	Add	78.9

cpu0

- Tracking the market is inherently ***sequential***
- Building a state machine

- E.g. spoofing feature might track a position
  - Watch Places & Fills on both sides over time,
  - Find large positions pressuring the market,
  - Then a cancel on one side,
  - Then ***reaping*** fills as market rebounds

Parallel Python

Sym	Time	Action	Price
NDAQ	1:23.456	Add	78.9
NDAQ	1:23.457	Add	79.0
NDAQ	1:23.458	Add	78.7
NDAQ	1:23.459	Add	78.9
NDAQ	1:23.459	Fill	78.7
NDAQ	1:23.461	Reject	78.9
NDAQ	1:23.463	Cancel	78.9
NDAQ	1:23.463	Add	78.9

cpu0

- Simple sequential Python
- Called with cluster as a simple array of rows

- Limited to what can be parallelized:
  - No global variables (function local only)
  - No native library callouts – unless thread safe
- Local self functions ok
- Most generic Python ok

DEMO!

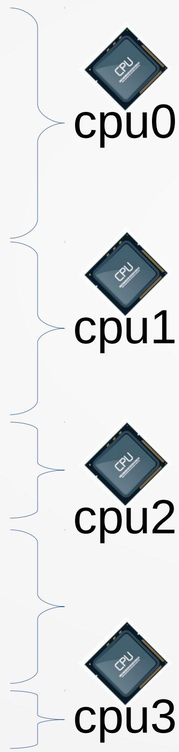
- Anonymized but real data

Policing The Stock Market with ML

Q&A

Parallel Python ML Modeling

Sym	Time	Action	Price
NDAQ	1:23.456	Add	78.9
NDAQ	1:23.457	Add	79.0
NDAQ	1:23.458	Add	78.7
NDAQ	1:23.459	Add	78.9
NDAQ	1:23.459	Fill	78.7
NDAQ	1:23.461	Reject	78.9
NDAQ	1:23.463	Cancel	78.9
NDAQ	1:23.463	Add	78.9
NDAQ	1:45.678	Fill	76.5
NDAQ	1:45.678	Add	76.5
NDAQ	1:45.679	Fill	78.9
NDAQ	1:45.680	Reject	78.9
NDAQ	1:45.680	Cancel	78.9
NDAQ	1:45.681	Add	78.9
NDAQ	1:55.681	Fill	78.9
NDAQ	1:55.681	Add	78.9
NDAQ	1:55.682	Add	78.9
NDAQ	1:55.683	Fill	78.9
AAPL	1:55.684	Reject	78.9
AAPL	1:55.684	Cancel	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Fill	78.9
AAPL	1:55.684	Add	78.9
AAPL	1:55.684	Add	78.9
AAPL	2:01.684	Add	78.9
AAPL	2:01.684	Add	78.9



- Most models in Python
  - Some in Java (H2O)
- Run sequentially per-cluster
- Clusters run in parallel
- Each CPU grabs a cluster, runs Python (Java) model, builds ML vectors, and scores for risk
- Work varies by model and cluster size – not uniform
  - Worklist load balances