accu 2023

ELECTRONIC TRADING FOR PROGRAMMERS

MATHIAS GAUNARD

Purpose of the talk

Finance? Trading?

- Finance sounds stuffy or unapproachable
- Part of the industry is secretive
- Disconnected from mainstream tech trends

Trading is very tech-driven

- Systematic and semi-system trading systems
- Low-latency execution
- Data analysis and modeling

$\Rightarrow$ Get engineers excited about the trading niche

C++ jobs in Trading

Back-office

- Core: general infrastructure, microservices, databases, orchestrators, etc.
- Connectivity: connect to trading venues or partners
- Modeling: use numerical methods to price assets

Mid-office

- Treasury: reconciliating and booking all positions, ensuring accounts are provisioned
- Risk: assess and characterize how much risk is tied to positions

Front-office

- Pricing: monetize models to find at which price to buy and sell
- Execution: place orders to execute trades at the best price

About the author

C++ Developer

- ~ 15 years
- C++ standards committee since 2011

High-performance computing

- Research in academia, software tools for parallel architectures
- Specialization in intra-CPU optimization
- Former owner of an optimization business (oil, aerospace, finance)

Trading

- ~ 7 years
- Market-making strategies
- Index, equity and FI options, cryptocurrencies
- Start-ups, banks and medium-sized organizations

Portofino Technologies

Low-latency connectivity, execution and microstructure

Outline

What is trading?

Exchanging assets with another party

- base/quote, often quote is a stable currency, e.g BTC/USD
- buy/sell base (sell/buy quote)
- usually done either as an investment or as speculation

Different ecosystems

- equities (stocks, indices on stocks, ETFs)
- fixed income (government bonds, corporate bonds)
- commodities (oil, metals, grain)
- currencies
- cryptocurrencies

How can I trade?

Financial instruments

- Trade assets outright (spot)
- Obtain a loan and trade against that loan (spot with margin account)
- Enter a contract with trade obligations at term (futures, perp swaps, CFD)
- Enter a contract with trade optionality at term (options, warrants)
- Smart contracts (blockchain-based enforcement)
- Exotic contracts (sophisticated legally-binding agreements)

Different products

- Listed on public exchanges
- Broker-dealer products
- Over-the-Counter only

Why trade?

Price move prediction (alpha)

- fundamental analysis of product (long term)
- events, news (medium term)
- market trends, statistics etc. (short term)

Connecting people

- arbitrage buy/sell flow
- collecting fees
- arbitrage different marketplaces
- arbitrage derivative instruments on same assets

Why electronically?

Larger pool of participants

- link venues across the world
- connect retail and professionals
- more competition, better prices

Transparency

- Records of all transactions
- Enforcement of due process
- MIFID compliance

Automation

- Enables looking at small opportunities a human wouldn't consider
- Systematic algorithms to run strategies consistently
- Low-touch enables higher volume

Who's trading?

Investors, buy-side

- Pension funds, mutual funds
- Venture capitalists
- Hedge funds
- Proprietary trading firms
- Retail

Trading services, sell-side

- Exchanges
- Market-makers
- Investment banks
- Brokers

How to interact

Direct, Over-the-Counter

- voice
- electronic, Request-for-Quote

Through marketplace/exchange

- best participant selected (usually anonymous)
- small fees
- multiple platforms
  - continuous, ”the screen”
  - auctions
  - multi-participant OTC-type platforms

Execution on-behalf

- finds best way to enter large positions over longer time periods
- larger fees
- methodology pre-agreed and/or performance-tracked
  - Flow traders
  - Algo-driven, vwap/twap

Outline

Vocabulary

- **bid**: buy order
- **ask**: sell order
- **offer**: ask
- **side**: whether it's buy or sell
- **tick**: increment of prices that are valid to place orders on
- prices p_1 **better** than p_2: $p_1 > p_2$ if bid, $p_1 < p_2$ if ask
- **BBO**: best bid and ask
- bid-ask **spread**: $best\_ask_{price} - best\_bid_{price}$
- bid and ask orders are **crossed**: $bid_{price} >= ask_{price}$
- **liquidity**: quantity addressable for trading, implied at good prices
- **touch**: liquidity close to the BBO

Continuous matching

Continuous

- Buyer/seller gets matched with sellers/buyers immediately if possible
  - If triggering match called **aggressor**, **taker** or **active** order
  - Removes matched liquidity, involved parties have traded
- Otherwise stays in order book and becomes **resting**, **maker** or **passive**
- Order book is always uncrossed
- Participants can amend/cancel their open orders

Limit orders

- Instrument identifier and side
- Maximum quantity (number of lots)
- Worst price per lot

Orders flags

- Immediate-or-Cancel
- Book-or-Cancel
- Icebergs

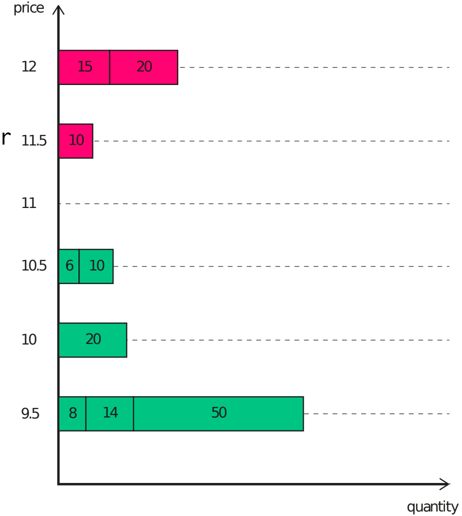
Order book, initial state

Order book

- Steady-state, all bids strictly less than asks
- Multiple orders per price level, arranged per insertion order (priority)

Example scenario

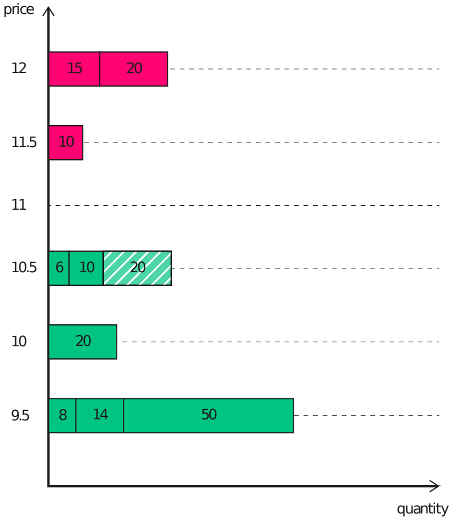
- Tick of 0.50
- Spread of 1, e.g. two ticks



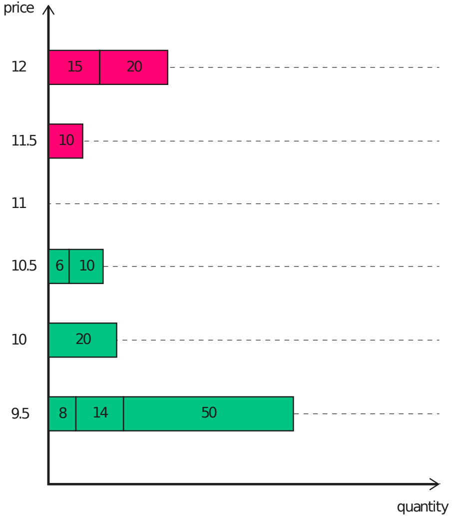
Insertion example, join

Buy 20@10.50

- Join the queue on best bid
- Spread unaffected, two ticks wide



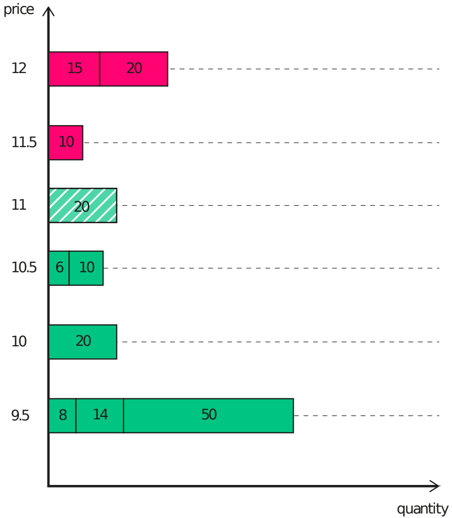
Order book, back to initial



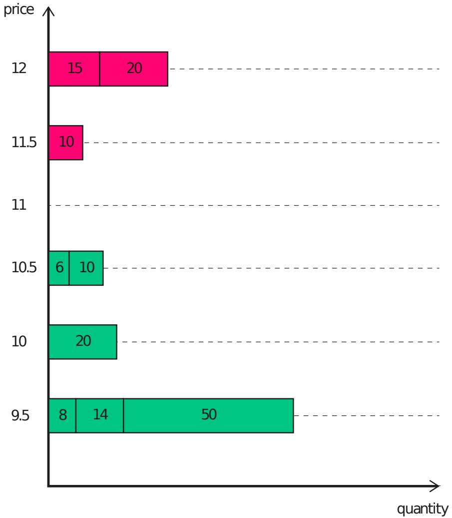
Insertion example, improve

Buy 20@11

- Establish new price level
- Front of the queue
- Spread tightened to one tick



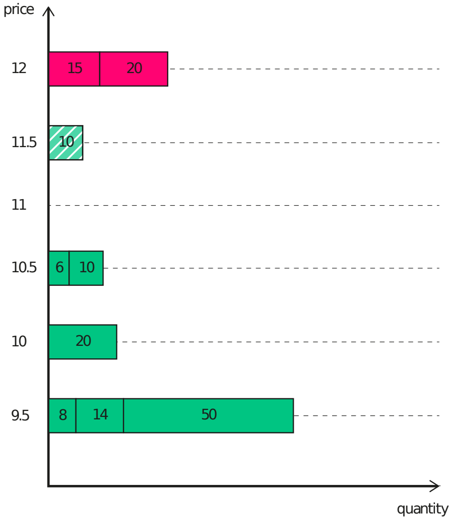
Order book, back to initial



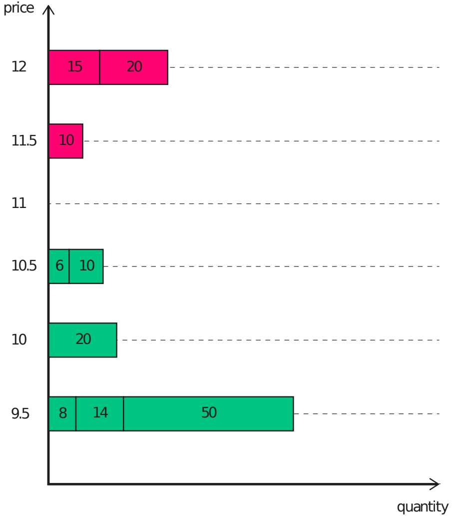
Insertion example, take and improve

Buy 20@11.50

- Trade 10@11.50
- Sell order disappears
- Establish new price level at 11.50 with remaining quantity
- Front of queue
- Spread unaffected, but market ”ticked up”



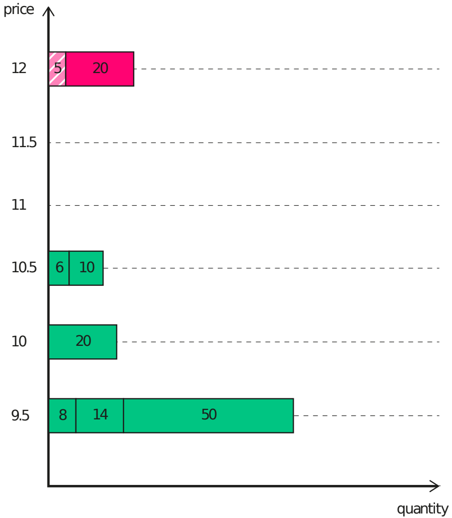
Order book, back to initial



Insertion example, take and widen

Buy 20@12

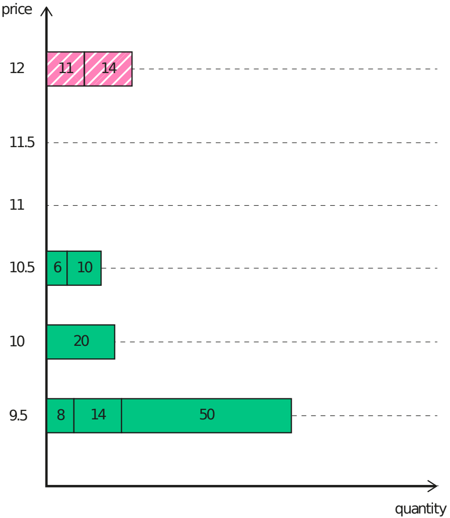
- Trade 10@11.50
- Sell order disappears
- Trade 10@12
- Buy order fully filled, not entering the book
- Sell order partially filled
- Spread widened, two ticks wide



Insertion example, pro-rata

Buy 20@12, different matching

- Pro-rata fills all participants relative to their participation to the total
- Different exchanges will provide different matching algorithms
- Can be a mix of different approaches



Order book, data structures

Index orders by identifier

- Support modify and cancel
- Hash table

Track priority of orders

- Linked-list of orders per level
- Ordered sequence of levels
  - Self-balancing binary tree
  - Circular buffer, dense tick representation

Many updates per second

- Pre-allocate and pool
- Hybrid (intrusive) data structures
- Optimize for operations close to touch
- Optimize hashing for indexing method of exchange

Outline

Matching conclusions

Active vs Passive

- Can execute instantly by crossing, but more expensive
- Can improve execution price by placing passive order, but must wait for other participant to cross

Passive execution probability

- Depends on queue position
- Can improve probability by establishing new price level, but worsens execution price

Market-making

Capture the spread

- Constantly buy and sell both sides passively
- Capture difference between bid and ask as profit
- Exposed to market moving before you can close the position

Aim for no risk

- No assumptions about price trends, do not go either position
- Bias system more or less aggressively or even cross to get out
- Build consolidated positions by aggregating cross-venue, statiscally etc.

Asset management

Alpha says whether to go long or short

- If you believe market will go up, buy before it does
- If you believe market will go down, sell before it does

Portfolio optimization

- Aggregate all price predictions across portfolio
- Find best set of trades to maximize portfolio value

Execution

- Could just cross, but expensive, and large executions move markets against you
- Smarter execution improves price, keeping more of the alpha

It's a spectrum

Alpha is good for best execution

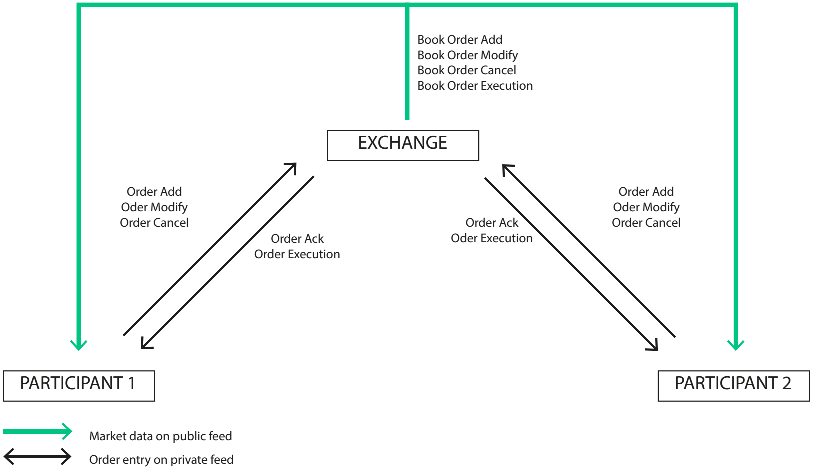
- Short-term alpha is crucial to make good execution decisions
- Medium-term alpha is good to avoid toxic flow (systematically one-sided executions)
- Long-term alpha is good to set target position

Execution is good to monetize alpha

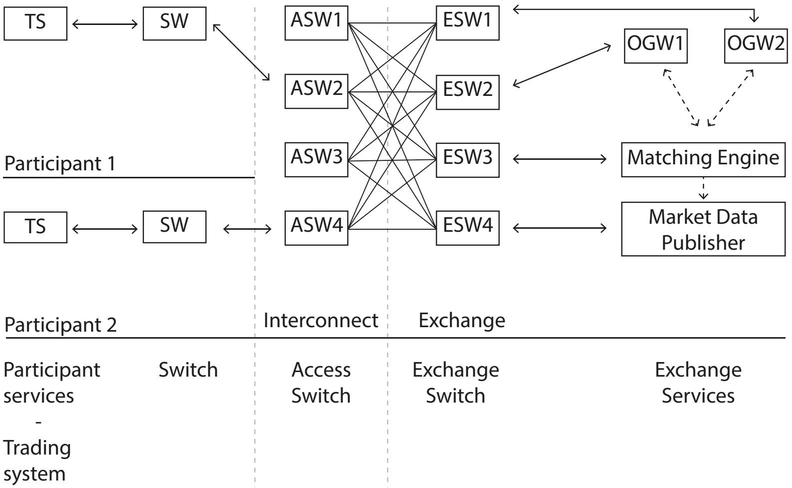
- Minimize market impact (position maintains expected value)
- Minimize slippage (cost of execution relative to a given trade decision)

Outline

Participant connections



Gateways and matching engine



Unicast vs Multicast

Unicast

- Send data for every single participant, bandwidth-hungry
- Goes through any router, including the open Internet
- No one gets it at the same time
- Can use TCP and have tailored per-participant data

Multicast

- Send data once, switches fan-out, bandwidth-efficient
  - Requires ability to propagate subscribers through network
- Participants get data at the same time
  - Modulo network congestion and ethernet signal phase
- UDP-only

Outline

Order book fidelity

Levels

- Level 1: BBO
- Level 2: aggregated quantity per price level
- Level 3: all individual orders

Netting

- Unnetted
- Throttled
- Coalesced

Recovery and reliability

Sequence numbers

- Out of order
- Gaps
- Incremental updates

Recovery

- Replay since beginning
- Snapshot

Serialization formats

Binary

- Flat, `reinterpret_cast`-friendly formats, e.g. SBE
- Delta encoding, e.g. FAST

Text

- FIX, key/value pairs
- JSON

Fragmentation

- Nice exchanges avoid it
- Others whatever IP says goes

Decimal numbers

Problem

- Often working with non-integral prices and quantities
- 0.1 cannot be represented exactly with binary floating-point
  Find $\{s, m, e\}$ such that $v \simeq (-1)^s \times m \times 2^e$, with $1 \leq m < 2$
- Approximations cause all sorts of problems

Decimal floating-point

- Find $\{s, m, e\}$ such that $v \simeq (-1)^s \times m \times 10^e$, with $1 \leq m < 10$
- IEEE754-2008, decimal32, decimal64, decimal128, backed by IBM and Intel
- Hardware support in POWER, software libraries
- Remains esoteric and slow

Decimal numbers (2)

Decimal fixed-point

- Fix the exponent and denormalize the mantissa
- Straightforward implementation, scaled integers
- Beware of operations that would change the scale
- No dynamic range/precision trade-off, beware of overflows

Fixed-point approaches

- Compile-time exponent, part of type
- Runtime exponent, schema that applies to a dataset
  - no redundant storage with all values
  - e.g. time series

Decimal numbers in practice

Exchanges

- Some use decimal floating-point (rare)
- Most use decimal fixed-point with negative exponent
  - Either same exponent for everything on protocol
  - Or per-instrument exponent
- Some just use decimal text

Recommendations

- Store data as integers, keep track of the relevant exponent they use
- Stick to scale-preserving operations when doing exact computations
- Switch in-and-out of `double` whenever doing non-exact numerical computations

Outline

Vocabulary

- **theo**: theoretical price from model
- **edge**: difference between theo and executed price, positive if profitable
- **credit**: minimum edge that we require to place/maintain orders
- **markout**: difference between mid/theo and executed price at different time horizons
- **slippage**: difference between the price you expect your execution strategy to be able to get and the price that you do get, positive if better than expected
- **greeks**: sensitivies of risk as a function to a particular price model input
- **hedge**: execute a trade on a given instrument to cover part or all of position in another instrument

Outline

Infer price from the order book

Micro-structure of the book

- Current state, how is liquidity spread
- Short-term state changes
- Statistics on state, what's ”normalcy”

Multiple features

- Imbalance between buyers and sellers
- Trends towards buying or selling
- Predict tick up/down

$\Rightarrow$ Build price at which to buy/sell, model credit on uncertainty and risk appetite of strategy

Tick size has a huge impact

- Big tick size, market is thick, one tick wide, lots of liquidity on BBO
- Small tick size, market is thin, spread out over multiple levels

Thin markets can provide better prices, but are more challenging to price correctly.

Exchanges aim to find a good balance so that price improvemnt is possible while still concentrating large liquidity at the top.

Simple BBO models

Formula

- Uncertainty is $ask_{price} - bid_{price}$
- Mid price: fair price is

$$bid_{price} + \frac{ask_{price} - bid_{price}}{2}$$

- Reverse-weighted sum: fair price is

$$\frac{bid_{price} ask_{qty} + ask_{price} bid_{qty}}{bid_{qty} + ask_{qty}}$$

Weaknesses

- Mid price doesn't account for quantity unbalance at all
- RWS behaves badly whenever BBO widens/tightens

Momentum

Accumulate over a window

- Find sensible signals that could impact future movements
- Parameterize them correctly to enable fitting
- Either binary decision or continuous price adjustments
- Window of effect and decay

Predict the future

- If you know what's going to happen, you can react earlier
- Signals can be driven by shape of network packet, grouping of updates, nature of order book updates, etc.

Correlations

Beta factor

- Instrument X value moves by *amount*
- Instrument Y value will probably move by β*amount*, with a certain likelihood, over a certain window of time

Applications

- Defend against possible moves, pre-emptively strike
- Hedge position in one illiquid instrument in a liquid one that it is heavily correlated to

Outline

Trading in a nutshell
- Exchange assets for profit
- Matching
- Strategy

Connectivity
- Feeds
- Protocol considerations

Modeling and pricing
- Generalities
- Microstructure
- Derivatives

Low-Latency Execution
- Why speed matters
- System architecture

Data analysis and research
- Data pipeline
- Research platform

Futures

Continuous-compounding formula

$F = Se^{r(T-t)}$

Cash flows beyond interest

- Dividends (equities)
- Storage costs (commodities)

Rate is unknown

- Bank rate generally known, but each participant may have a better rate
- Infer the rate (or rate offset) from the market
- Future vs spot, expiries vs each other

Options

European, call, continuous-compounding

$$C = N(d_1)S - N(d_2)Ke^{r(T-t)}$$

$$d_1 = \frac{\ln \frac{S}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}$$

$$d_2 = d_1 - \sigma\sqrt{T-t}$$

Fitting

- Need to infer both r and σ
- Want to ensure σ has a certain smooth structure across K and T
- Fit a smooth model until you minimize error with observations

Greeks

- Delta: sensitivity to changes in S
- Vega (kappa): sensitivity to changes in σ
- Theta: sensitivity to changes in t
- Rho: sensitivity to changes in r

Options (2)

American options

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV \leq 0$$

Use an approximation or solve the PDE

Conflicting pricing

Multiple options

- Price future from spot, price future implicitly
- Price spot from future, price spot implicitly
- Blend the two, etc.

Change and arbitrage

- If markets efficient, all prices agree
- In practice more/less uncertainty on certain markets
- Some patterns in who moves first

Outline

Asynchronous change requests

Constant stream of updates

- Many participants
- Orders being added/modified/cancelled multiple times per millisecond
- Exchange has to distribute a lot of data, may be slow
- May even be throttled on its way to you

Join the queue

- Observe market in a given state, want to affect a change
- Change request submitted and queued
- Exchange drains the queue of changes, eventually processes it
- Observe market with your change applied, tens of milliseconds might have passed

Fairness

Access to information

- Preferential access
- Does network provide information to all participants at the same time

Order processing ordering

- Does everyone go through the same gateway
- Is there any reordering on the way to this gateway or internally

Special rules make speed less of a concern

- Micro-auctions
- Asymmetric delays
- Pro-rata matching

Determinism

Deterministic

- Fastest always wins
- Clear, fair, efficient
- Technologically more challenging for exchange
- Leads to people building FPGAs and ASICs

Non-deterministic

- Fastest only has an advantage, but semi-random
- Exchange can have dynamic behaviour based on load
- Leads to reverse-engineering and finding whatever can be gamified to improve the odds
- Mostly lots of tricks but no need for hardware

Pick-offs

Theoretical price-driven

- Only willing to buy for less/sell for more than some price model
- Market conditions change, theo changes, you cancel
- Aggressor is faster, bad trade

Risk-driven

- Have lots of orders on many instruments or different venues
- Don't necessarily want them to be filled all at once
- One order is filled, cancel other ones
- Aggressor is faster, overtrade

Achieve good priority

Bad priority $\Rightarrow$ never get to trade

Price improvement opportunities

- Event-driven and competitive
- Tied to price move predictions

Proactively place orders where price might move to

- More load and complexity
- Additional risk in the book

Everyone wants to take the good stuff

New order crosses theo

- Spread tightened, new price attractive to your theo
- Try and send an order to match against it
- Other participants might do the same, fastest takes it

Event causes theo change

- Something happens that affects your price model (usually other market)
- Now you think many existing orders are mispriced
- Try and send orders to match against all of them
- Other participants might do the same, may only get some of it

Some metrics

Fill rate

- How often do I get filled before cancelling
- Relative to various priority metrics

Hitting rate

- How often do I get anything when I try to take liquidity
- Amount of credit/edge
- Latency of sending the message through

Time on the wire

Trigger to order

- When was network packet triggering decision received
- When was network packet with order emitted out

Streaming at 10Gbps

- MTU of 1500 is 1200ns
- End of Frame to End of Frame
- Start of Frame to Start of Frame
- Start of Trigger to Start of Frame

Software times

Not whole picture

- Delay between packet received and picked up by software
- Delay between sending packet and it being serialized out

Low-latency measurements

- `rdtsc(p)`
- `rdpmc`

Meaningful statistics

Quantiles

- Min – how fast you can expect to get
- Median – how fast you are typically
- 90^{th} percentile – how well are you protected
- 99^{th} percentile – how bad can it get

Determinism

- Removing tails is hard, but important
- Important events happen rarely
- Control flow divergence increases jitter

Outline

Soft real-time constraints

By order of importance

- No system calls that block non-deterministically
- Real-time scheduling guarantees and affinity pinning
- Lock-free synchronization (i.e. no blocking system calls at all)
- No memory allocations
- No system calls at all
- Wait-free synchronization
- Limited code flow divergence (i.e. deterministic execution time)

Hybrid trading systems

Special-purpose

- Super fast cancel
  - Better cancel more often than needed
  - The simpler, the easier it is to make it fast
- Fast hitting
- Slower quoting, relaxed coding constraints

Integrated

- More precise cancelling
- Enhanced capabilities for sophisticated stategies

Ethernet, IP, TCP

TCP – byte stream

- Handshake
- Ack window
- Nagle algorithm
- Retransmissions

IP – packets

- Routing
- MTU and fragmentation

Ethernet – frames

- Bandwidth
- Signal phase

Low-latency networking

Kernel-bypass

- Direct communication with Network Interface Adapter in userland
- DMA, write-combining memory, PCI-Express
- Disable interrupts, too slow to read means dropped data

Userland TCP/IP

- Receive/send ethernet frames or frame fragments in application
- Re-implement all of TCP and IP without relying on the kernel
- Shortcuts for reliable networks

Ready-made solutions

Solarflare (now Xilinx)

- OpenOnload, BSD socket compatibility layer, high conformance
- EFVI, low-level API
- Onboard FPGA since Xilinx acquisition

Exablaze (now Cisco)

- exanic low-level API, pre-loading capabilities
- Onboard FPGA as core of the system
- exasock, BSD socket compatiblity layer, not-quite-conforming

Others

- Mellanox (now Nvidia), more targeted at HPC and Infiniband
- Myricom, other HPC pioneer, notable for a Windows API, now defunct

Standard approaches

DPDK

- Linux foundation
- Large userland framework for kernel-bypass networking
- Supports traditional NICs (Intel, etc.)
- Userland TCP implementations

`io_uring`

- Linux kernel
- paradigm shift removing system calls
- new languages (e.g. Rust) building their networking around it

Threading model

Few threads

- No reliance on OS thread scheduling
- Cooperative scheduling intra-thread

Share-nothing

- Objects local to a given thread
- Communication via lock-free queues
- Seq-locks for state sampling
  - recently made compatible with C++ memory model

Some frameworks

- Asio, not the best fit
- Seastar, good principles

State of the union

Race to the bottom

- Normal software $< 10ms$
- Software with real-time in mind $< 100\mu s$ – sweet spot?
- Ultra low-latency software $< 3\mu s$
- Normal FPGA solution $< 500ns$
- Ultra low-latency FPGA solution $< 50ns$
- ASIC $< 30ns$
- Above and beyond $< 10ns$

Outline

Recording time series

Type of inputs

- Feeds
- Pricing infromation
- System internals
- Trade events

Requirements

- Hundreds of GBs to TBs per day
- Must not impact latency of trading
- Data needs to be available within hours

Serialization and formats

- Binary for scalability
- Schema-driven to avoid redundant data
- Compression for smaller bandwidth usage
- Streamable for resilience

BSON, no schema, row-by-row

Flatbuffers, SBE, schema, row-by-row

Arrow IPC, schema, batches of rows, column-by-column

Recording metrics

Type of inputs

- Amount of data received/processed, load
- Latency of a sub-path
- Other statistics

Requirements

- Don't need all state transitions
- Infrequent sampling
- Moderate amount of data
- Must not impact latency of trading
- Data needs to be available within minutes

Interaction model

- Database polling for new state better than pushing state changes to database
- Online stats can be done over polling window then recorded as a sample in time series

Prometheus, example of good solution

Ingesting and working with data

Databases

- Flat arrow/parquet files
- Spark, Athena
- KDB
- InfluxDB
- MongoDB

Data manipulation

- Python with pandas
- Q
- R
- MATLAB

Data processing at scale

- Parallelization: Dask, Ray
- Orchestration: Prefect, Airflow

Compute on demand

- Dedicated cluster
- Cloud spot instances on-demand

Outline

Quality metrics

Price

- Estimate predictive power at different horizons
- Assess if systematically biased

$\Rightarrow$ Markouts the main tool.

Execution

- Taking: mostly a question of speed
- Making: mostly a question of queue priority

$\Rightarrow$ Simulate, find what you can take / when your queued orders would eventually get filled

Simulation

Simulation matching

- Replay historical market data
- Run trading engine placing fake simulation order
- Overlay or merge simulated orders on top of historical ones
- Fill simulated order when would be matched against historical order

Problems

- Market impact
- Mismatched historical/simulation order books
- Modeling exchange delays

Network simulation

Fake exchange

- Re-implement exchange with same network protocol
- Hooked up to historical data and simulation matching engine
- Instrument the network to be able to make it look like it's the actual exchange

System testing

- Can run the actual binary that goes into prod
- Measure actual system latency
- Thoroughly test the system

Optimization

Simulate all the options

- Explore the universe of possibles
- Find what features are significant
- Find which combination works best

Offline validation

- Make sure the system works before going to product
- Know what performance to expect

Questions?