SJTU Project

- Algorithm Trading

4th Sept, 2016

Agenda

1. Algorithm Trading
2. VWAP/Iceberg/SORT
3. Project
4. Q & A

1. Algorithmic Trading

Why algorithmic trading ?

It’s all about…

Algorithmic Trading?

If look for a definition, you will find …

"the use of electronic platforms for entering trading orders with an algorithm which executes pre-programmed trading instructions whose variables may include timing, price, or quantity of the order, or in many cases initiating the order by a "robot", without human intervention"

Source: https://en.wikipedia.org/wiki/Algorithmic_trading

"a computerized model that incorporates the steps to trade an order in a specific way"

Source: Algorithmic Trading & DMA, Barry Johnson 2010

A typical algorithm order

```
qty      = 6,000 shares
security = google inc
Side     = we buy the security
```

Buying the shares		
time	quantity	Offer price
9:30:00	1000	884.9
9:35:00	1000	887.8
9:40:00	1000	892.3
9:45:00	1000	889.6
9:50:00	1000	891.0
9:55:00	1000	886.2

- **notional** amount of $5,331,800.0

5,331,800.0 (888.6 avg/share)
+
Fees + commissions + taxes
+
?

- **known before trading**: fees, commissions, taxes, user strategy, urgency

- **unknown:** market impact, timing risk, delay, opportunity, trend costs, price move
  these are unknown until we make a trade and/or the trading session is over

Let’s make it simple

Why not trade the full qty right away ?

- If we could predict the future we wouldn’t be here ☺ .
  - Price can be more favorable later
  - In the absence of alpha / view in the market it is better to minimize trading cost.
- We cause market impact by crossing the spread with a large quantity

“crossing the spread”: no matter on which side we are, if we want to trade we need to Match the price on the other side, effectively “crossing” the spread between them to reach the opposite side and trade.

The resulting cost, is the spread value itself: 2.4$

One big question: how much did these trades cost us ?

Market impact costs

time	Offer price	# shares
9:35:00	887.8	4,000
~~9:35:01~~	~~884.2~~	~~3,000~~
9:35:02	882.3	8,000
9:35:03	882.3	5,000
9:35:04	882.3	3,500
…	…	
9:40:00	892.3	2,000

"We have a buyer"

time	Share price	# shares
9:35:00	887.8	4,000
9:35:02	885.3	2,000
9:35:03	886.2	2,000
9:35:04	885.1	1,500
…	…	
9:40:00	893.3	5,000

Market reacts: we wiped out a price level in a single trade

We moved the market: prices and quantity move away from us

Our future trades will cost more, nuking the savings from spliting the order

we want the savings of volume/price opportunities without moving the market

Market impact costs

time	Offer price	# shares
9:35:00	887.8	4,000
9:35:01	884.2	1,600
9:35:02	882.3	8,000
9:35:03	882.3	5,000
9:35:04	882.3	3,500
…	…	
9:40:00	892.3	2,000

We take only a portion of displayed qty:
ratio can be guessed from previous days market activity
→ historical data, crunching numbers

Spread costs

time	offer price	# shares
9:35:00	887.8	4,000
9:35:01	884.2	1,000
9:36:02	882.3	8,000
9:36:53	882.3	5,000
9:38:04	882.3	3,500
…	…	
9:40:05	895.0	1,000
…	…	…
9:45:13	894.8	1,000
…	…	…
9:51:02	894.9	1,000
…	…	…
9:57:33	894.9	1,000
…	…	…

waiting for 5s between trades:
Missing opportunities

Each time we trade, we “cross the spread” and pay
The spread costs: instead of taking current price we
Could have waited passively for a better price

We pay for urgency

Optimization
We can optimize our placements: while waiting, place
An order with a price better than our current price
Stay there waiting for someone to hit our bid for the
Period between 2 spread crosses.

Spread costs

time	offer price	# shares
9:35:00	887.8	4,000
9:35:01	884.2	3,600
9:36:02	882.3	8,000
9:36:53	882.3	5,000
9:38:04	882.3	3,500
…	…	
9:40:05	895.0	1,000
…	…	…
9:45:13	894.8	1,000
…	…	…
9:51:02	894.9	1,000
…	…	…
9:57:33	894.9	1,000
…	…	…

We split trading into 2 phases: aggressive, and passive
- aggressive at 9:35:01, we cross the spread
- passive at 9:35:03, we place an order @882.3

Note that even passive placements can move the market
Same strategy as active placements: use historical data to infer size, time and price for the passive placement

Let’s pause for a second

What do we get so far … ?

An algorithm that does some trading

- tries to have low market impact:
  - slices the original qty on a longer period of time
  - when taking quantity, takes a portion of it, not all of it
- try to reduce spread costs, place orders passively, becoming active so the algorithm makes progress toward completion when passive fails
- use historical data to optimize orders placements
- vocabulary

- a trading schedule is followed by the algorithm
- fill the order by following the schedule by providing passive/active executions

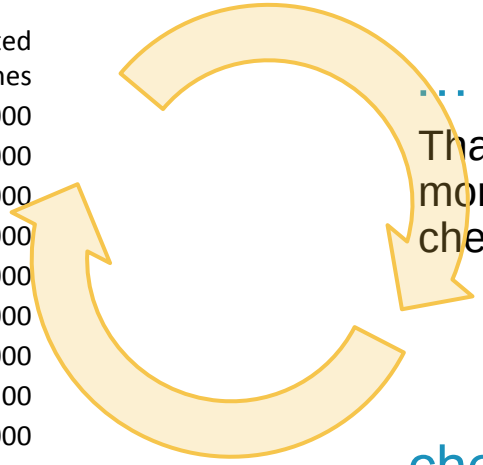
Basis to understand Benchmarks

Tracking a benchmark: VWAP (more)

We need a plan …

Or more precisely, we need a schedule, to ease tracking:
Get historical data / volume distribution for the security, compute expected volumes/placement

	Expected volumes
9:30	24000
10:00	20000
10:30	9000
11:00	6000
11:30	3000
12:00	3000
12:30	3000
13:00	3500
13:30	3000
14:00	3000
14:30	3000
15:00	9000
15:30	10000
16:00	15000
16:30	30000



… then hen we need to execute it

That means trading: we start passively, then more aggressively as we reach schedule checkpoints

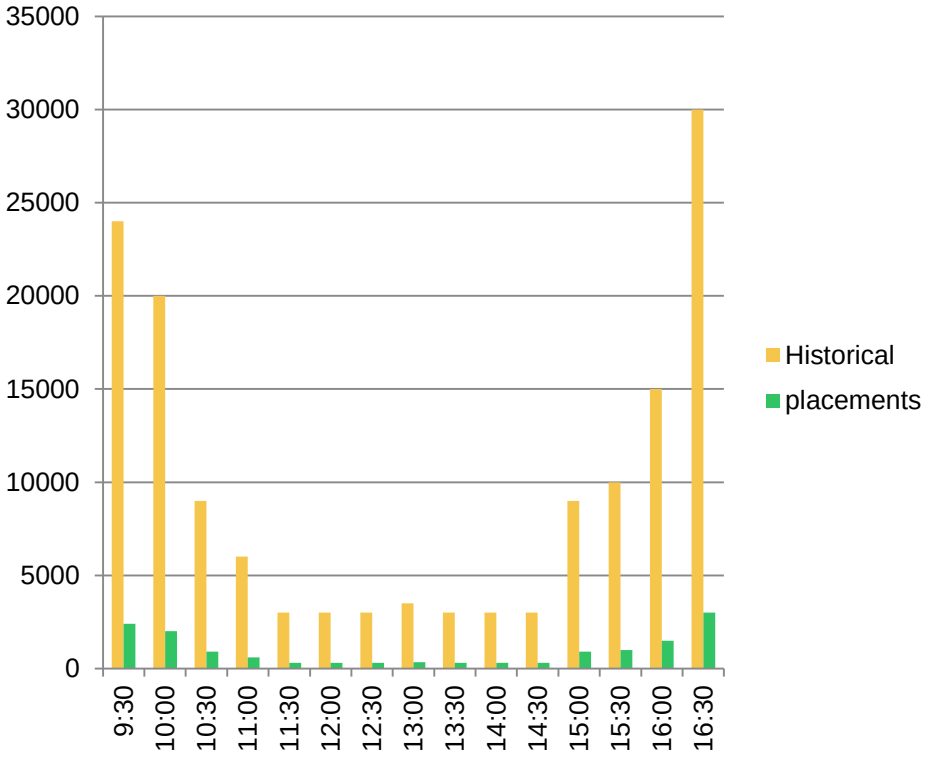
… check that the plan still holds …

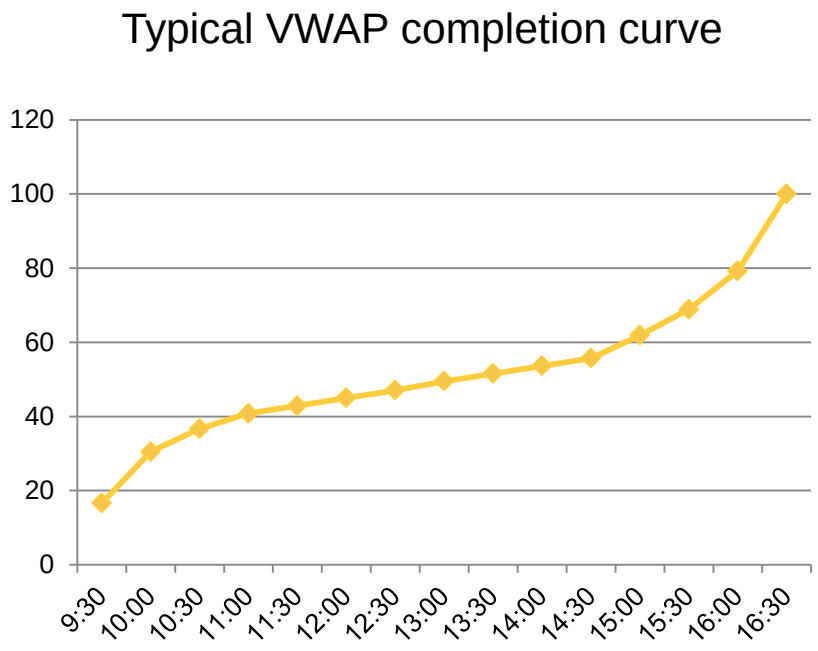
Crunch market data/events and evaluate if given the new market conditions we need a New plan

Tracking a benchmark: Volumn Weighted Average Price

Tracking is usually done by using historical volumes per Ts
This gives us a distribution of volumes across the day

- Placements will follow the distribution
- qty we want to trade is spread through the day to avoid market impact
- Hence if there is some timing risk, avoid using it





Iceberg Orders

ICEBERG!

IBM on NYSE

Bids			Asks		
100	[500]	140.05	100	[500]	140.07
100	[700]	140.04	100	[700]	140.08
100	[200]	140.03	100	[300]	140.09

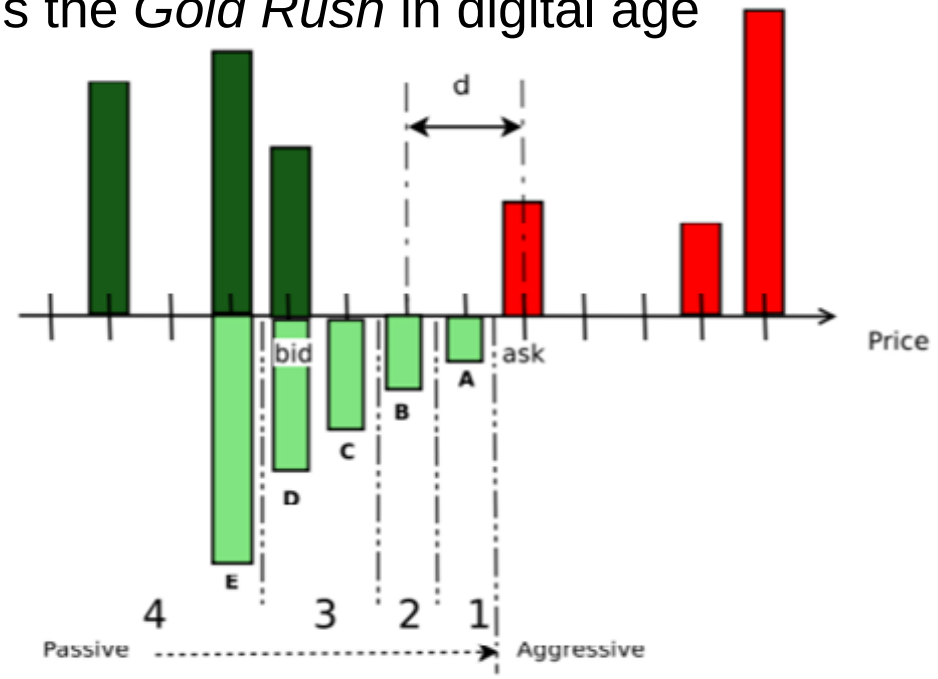
Exchanges Support 'Iceberg' Orders:

- Part of the order is displayed
- The rest is in 'reserve' and not displayed
- In the above example, NYSE's inside bid is showing 100 shares, but a total of 600 (100+500 reserve) is available to trade.

Iceberg Orders

Customize
- Exchange support is limited (and sometimes a paid service)
- How to customize an iceberg order strategy for your clients

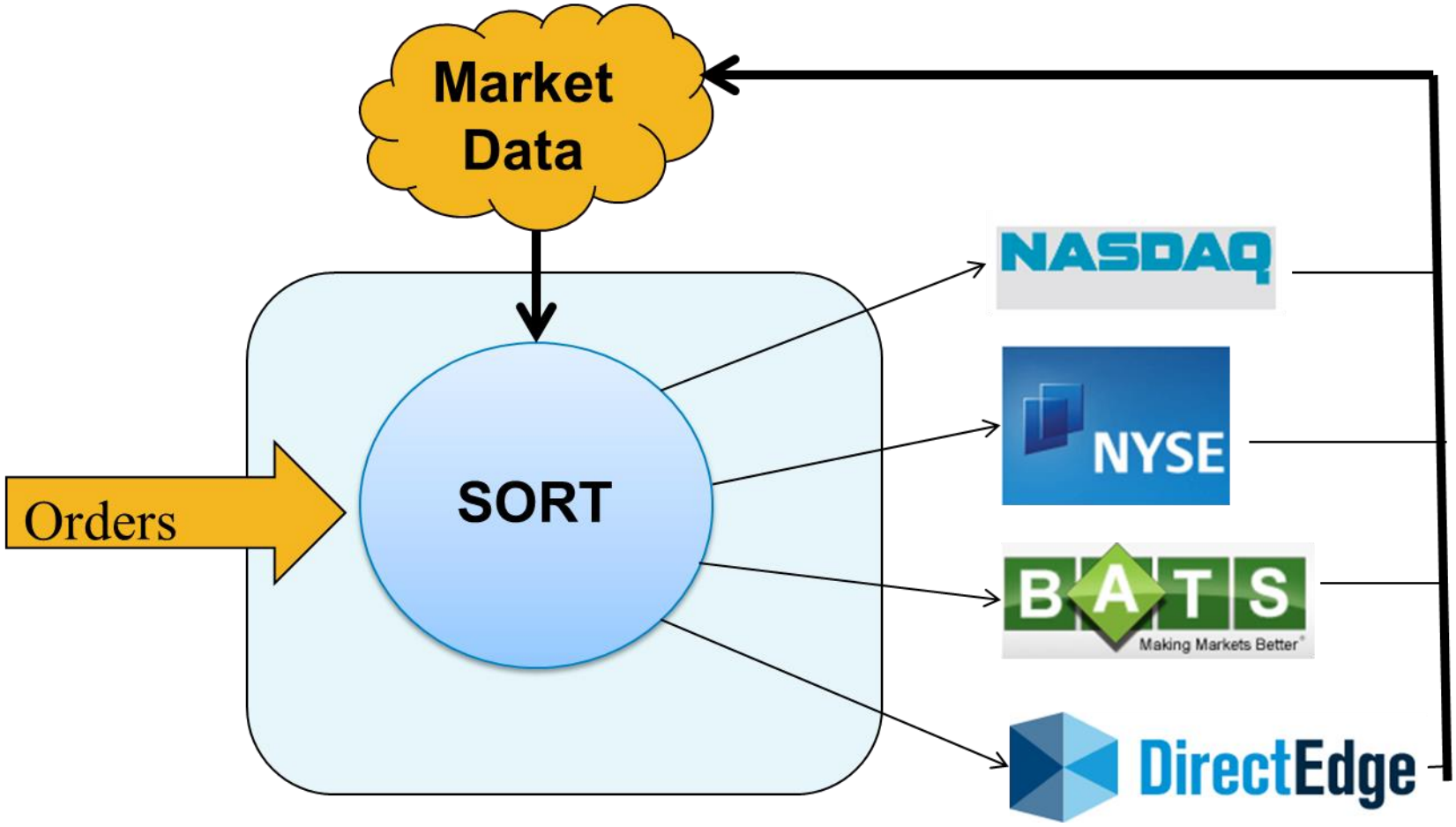
Detection
- Don’t hit near side iceberg !
- Far side Iceberg is liquidity !
- Iceberg Rush is the *Gold Rush* in digital age



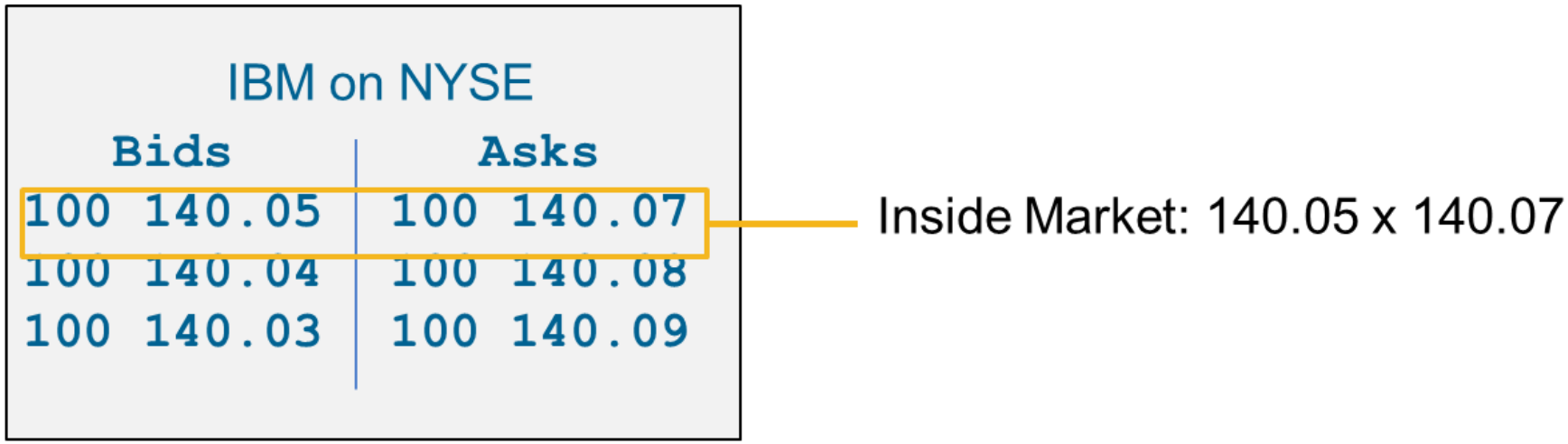
http://www.tinbergen.nl/~sofie2012/papers/HautschHuang2012.pdf

2. SORT Brief

SORT: Smart Order Router



SORT: Order Book



- When an exchange receives an order, 2 things can happen:
    1. Order is immediately executed
    2. Order is placed in the order book

SORT: Consolidate Books

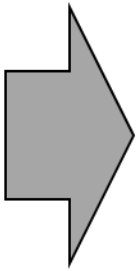
Consolidated Order Book

IBM on NYSE

Bids	Asks
100 140.05	100 140.07
100 140.04	100 140.08
100 140.03	100 140.09

IBM on Nasdaq

Bids	Asks
200 140.04	200 140.06
200 140.03	200 140.07
200 140.02	200 140.08



Consolidated IBM Book

Bids	Asks
NYSE 100 140.05	NSDQ 200 140.06
NYSE 100 140.04	NSDQ 200 140.07
NSDQ 200 140.04	NYSE 100 140.07
NYSE 100 140.03	NSDQ 200 140.08
NSDQ 200 140.03	NYSE 100 140.08
NSDQ 200 140.02	NYSE 100 140.09

SORT: Speed Matters

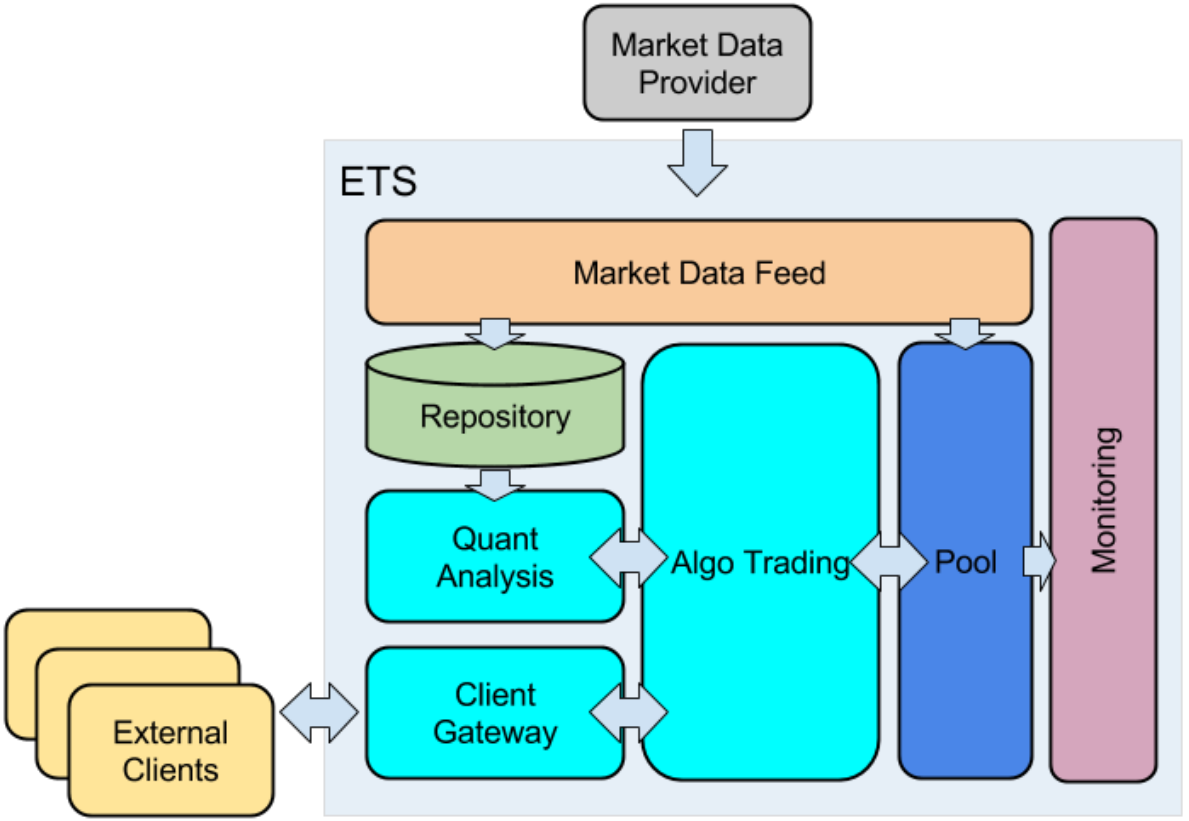
- When we are waiting for market to come closer, can we grab the opportunity in the sub-millisecond window?
- If you are fast enough, you are seeing future.

3. Project

Goals

Build algo trading platform which have the following major components:

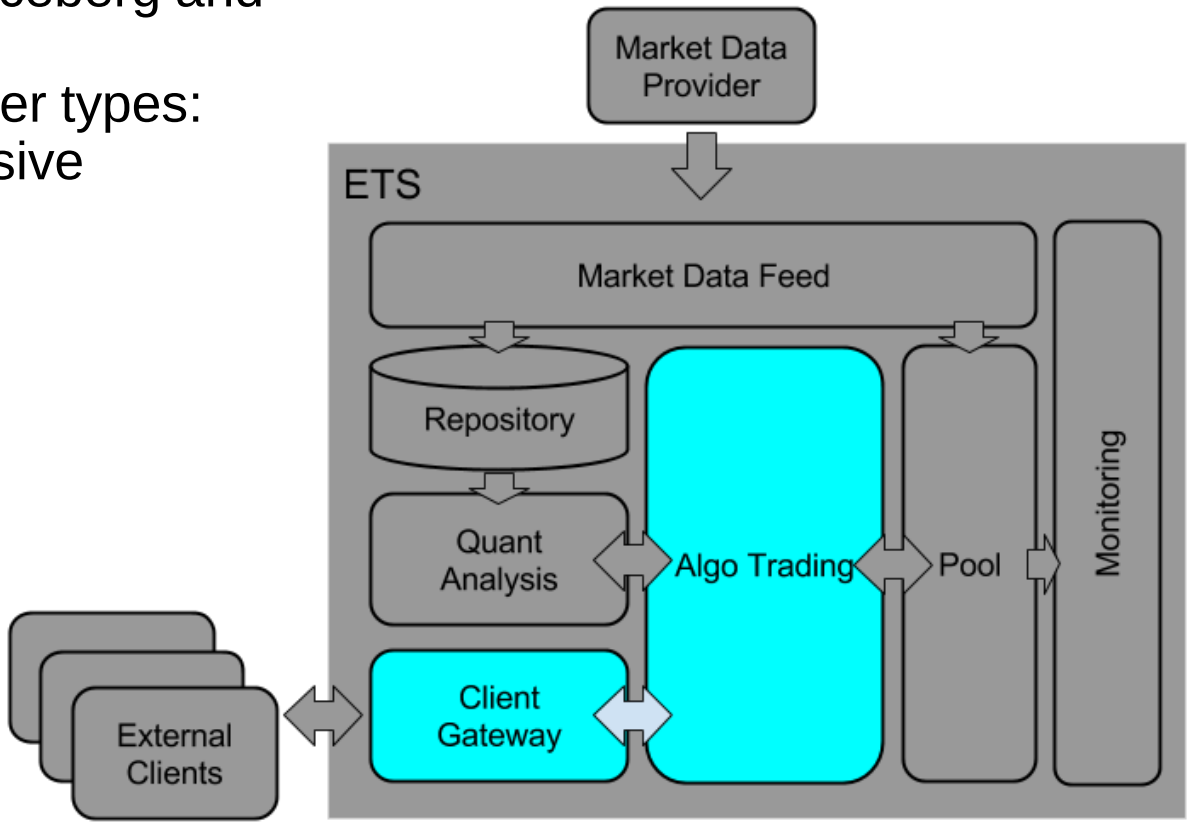
- Market Data Reader
- Algo Trading Engine



Goals

Algo Trading

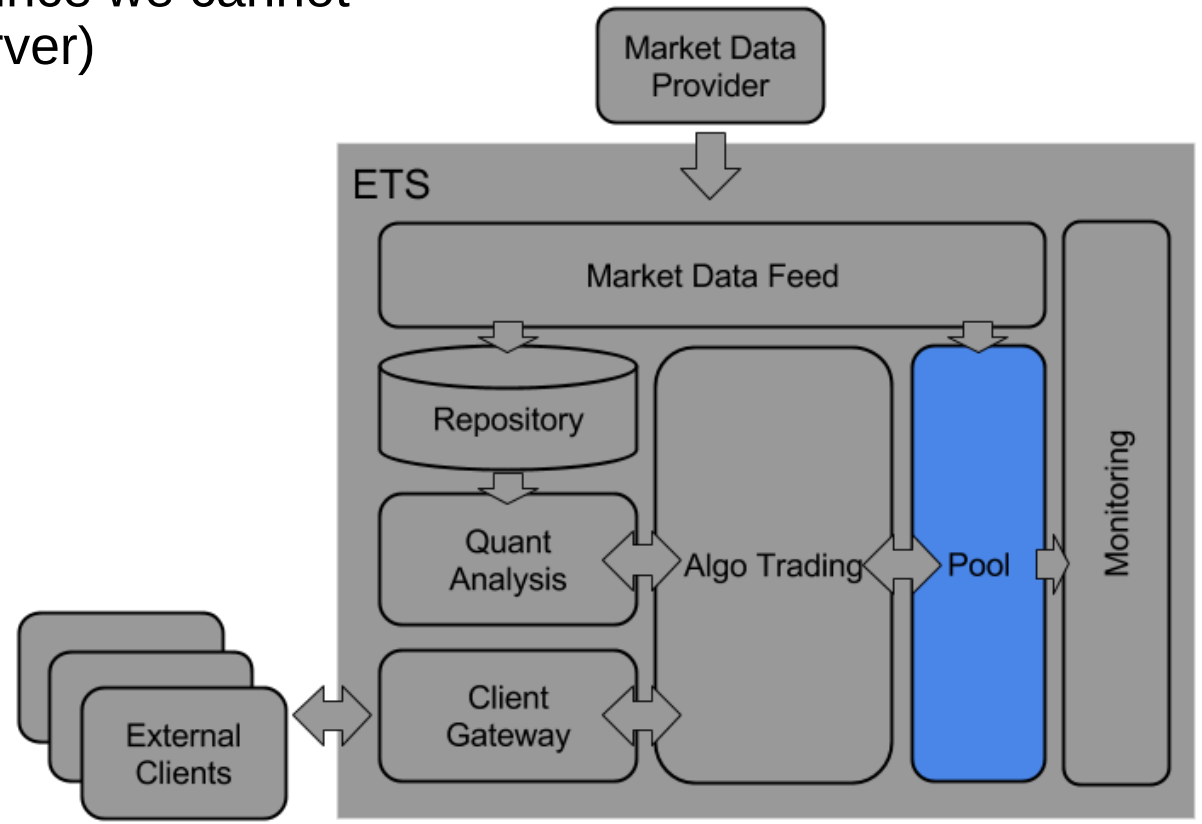
- Framework to host different types of algorithms
- Under the framework, it supports 2 specific algorithms: Iceberg and SORT
- Support different order types: Aggressive and Passive
- Work at minute level
- Client connectivity



Goals

Pool

- Get the realtime/historical market data
- Validate the orders
- Match the orders (since we cannot connect the real server)



More Details

Basic

- Build market data feed and storage to extract 2 or 3 stock prices
- Build quant analysis engine to generate trading schedules for VWAP
- Build a consolidate book for SORT
- Build algo trading engine to execute orders
  - Orders can be created via command line
  - Monitor the order execution flow via algo trading log output
  - Automatically select the destination based on consolidate order book

Advanced

- Chain TWAP and SORT together
- Enhance SORT engine to send order only when market is tradeable
- Reduce tick-to-trade latency to micro-second level

Technology stack and references:

Programming language: Java/Python (recommended)

References :

- **Algorithmic Trading & DMA** by Barry Johnson
- **Inside the Black Box** by Rishi K. Narang
- **Quantitative Trading** by Ernest Chan

Q & A?

Shoot !