I’ve worked for four years developing market making and arbitrage bots at the brazilian futures exchange, so I wanted to share some insights here that may be useful for the MEV auction debate.

Probably the most profitable HFT strategy is a relatively simple one that I call the ‘unidirectional arbitrage’. The way it works is that you see some large trade moving the price of some very liquid market, and then you copy that price movement in all other markets that are equivalent or highly correlated with it.

For example, if you see one big trade moving up the mini-S&P futures you can do all of the following:

- Buy micro-S&P futures
- Buy all stocks that are highly-correlated with the S&P (at NYSE and NASDAQ)
- Buy ETFs that follow similar indexes
- Buy call options for the S&P and for stocks
- Sell put options for the S&P and for stocks

Of course, none of this is guaranteed a profit, as the prices may fall before you close your position, but given that you will be doing thousands of trades per day, you really just need to win on average, which you definitely will if you can manage to do your trades before everyone else can.

There are basically three ways of having your trades execute first.

1. If the exchange you are trading at has a “fair” or a “first-in-first-out” ordering policy, then you need to be really

fast. If you are trading in two markets in the same exchange then you may need FPGAs, overclocked CPUs, special code optimization, expensive network interfaces, and so on. For different exchanges you will need straight-line submarine cables, microwave towers, or maybe these new hollow-core fibers. There may also be some tricks to get the information first, such as placing small orders in the market to let you know about big trades before everyone else.

1. If ordering has some random element, then you need to send tens or hundreds of orders and hope that one of them will execute. This may happen accidentaly, for instance, if the exchange has several gateways with multiple processes running. The more orders you send, the higher the chances that you’ll get lucky with the process scheduling.
2. If instead you have to pay to get a better order endpoint or to get the trade information faster, then you have to pay it. While in principle some exchange could auction the fastest connection for a single highest bidder, I’ve never seen an exchange to that, so in general, you still need techniques from (1) and (2) to win.

Of course, in practice there is often a mix of the three ordering strategies (you can pay for colocation, lower latency is usually first, but there is some random component). Sometimes there are also some other tricks that only insiders know and that work only in some exchanges.

While this is my experience with traditional finance, HFT in crypto probably works exactly the same in the centralized exchanges. In DeFi of course it is totally different because the orders are all in the mempool to be seen and can be frontrunned, sandwitched, and you can even place and remove huge amounts of liquidity from the pools to extract all the trading fees.

These transaction reordering exploits are so hurtful right now that many are arguing that latency arbitrage is not a problem by itself, because of some positive consequences of arbitraging such as allowing fast liquidations and propagating price information.

As an insider that made quite some money by “propagating information” a few microsseconds earlier than the others, I disagree. All revenue made by arbitrators has to come out of somewhere, and where it usually comes from is from the liquidity providers.

In centralized exchanges, the money comes out of the market makers who respond with higher spreads. The exchanges don’t like the way the orderbook looks with these high spreads, so they create subsidies with negative fees (maker rebates). Of course, they have to charge the high “taker fees” to pay for it.

In decentralized exchanges, the arbitrage revenue come out disguised as very high impermanent loss

, which people seemed to have regarded as a normal fact of life than can only be compensated by high trading fees or liquidity mining, which by the way is nothing but another type of subsidy.

Yes, traders do need to be rewarded for incorporating information into the markets, and statistical arbitrage needs to be rewarded, but only inasmuch as it is the information incorporated is really unique. One shouldn’t be paid to ‘incorporate’ obvious information a few microsseconds before someone else would. A good description of the problem is given by Budish et al.

The proposed solutions for transaction ordering in DeFi mirrors the three ordering components mentioned above:

1. You can propose some type of first-in-first-out mechanism in which transactions sent first arrive first.

Not only this is a very hard task in decentralized setting, the goal may even be undefined! This may be a bit theoretical, but at least according to general relativity you can't even talk meaningfully about ordering events that happen at different positions in space. If one could agree on what exactly is the goal here, this might help alleviate frontrunning, but would probably add a lot of complexity to the consensus layer and would still result in negative-sum latency arbitrage wars as in traditional HFT.

1. You can propose some sort of random ordering of transactions as@pmcgoohan appears to have done here and as others have tried to propose before.

As hard as it may be to get random ordering in a 'fair' way in which the seed cannot be manipulated by the miners/validations and so on, it doesn't change the fact this will encourage traders to sometimes send as many orders as they can to execute first.

One may think that, because each order always has a 50% chance of being placed after any other, there is no money to be made with frontrunning or sandwitching attacks, but that is totally false. There are many clever ways to send many transactions and have the wrong ones fail. One can even deploy a smart contract that inspects a given user's balance and the uniswap price to decide whether any order relayed though it is to proceed or not! So I don't think random ordering should be involved in any proposed solution.

1. You can have MEV auctions as proposed by @vbuterin here

This approach has a very significant disadvantage in that, by allowing MEV to be extracted more efficiently, it will make the frontrunning costs paid by regular users of uniswap-like exchanges even worse than they already are.

Even so, I'm strongly in favor of that proposal

. First because it changes consensus rules at lot less, second because at least resources are sent to validators instead of being destroyed as with transaction bloating or useless underground cables, and third but not less importantly, because the frontrunning costs are temporary and can be solved completely in the application layer

How is this so? The key is that, while blockchains are great for decentralized smart settlement, they are not necessarily good for time-critical trading. On-chain order matching is not necessary

for decentralized custody and settlement, which is what makes DeFi great. Disclaimer: I'm currently building a hybrid exchange with off-chain batch-auction order matching.

I will not go into details of how this can be done now, but one can create a TradingBalance contract that is similar to a payment channel, but in which users can not only sign messages authorizing payments, but also messages authorizing specific swaps (e.g. I authorize you to exchange up to 1000 DAI into WETH at a maximum price of 3000$ as long as you settle this trade on-chain before block X). Smart contracts could do a similar authorization by a function call.

If the exchange is malicious, it can of course frontrun the order itself, but here the exchange has a reputation to keep. The exchange could give a reply back in milisseconds confirming the trade with a signed message. Again, the exchange would have the power to revert the trade until it is settled, but its reputation is on the line, and because it can extract trading fees from users who trust it, this reputation will be worth much more than the profits that reverting a few trades might bring.

So I think this explains how I believe application layer solutons can be designed to remove frontrunning costs while settling them on-chain later.

What I have not explained yet is how an exchange can prevent latency arbitrage from happening. I've actually been working for the last three years in a solution to this problem. I developed a design for a new matching mechanism based on batch auctions, but this is one without uniform pricing

. We instead design it so it can have a truthful property for the liquidity takers (they don't need to worry about their own price impact). We use a cyclical mechanism in which each step is inspired by Ausubel (2004).

We have a demo working here, and we are testing it in a centralized exchange setting. It will look just like a regular orderbook (that is the idea), but one can place an order on the book and the order might be executed with price improvement later. For example, suppose there is an order selling for $100 initially at our exchange, the price rises in some other exchange and three bots send buy orders at $108, $109, $110 respecively at roughly the same time. Then the last bot will get to buy it and will pay $109 (like a second-price auction), and the selling order will get price improvement. Early documentation is here.

Gnosis team @mkoeppelmann, @fleupold and others, I know you are working on a different batch auction mechanism with CowSwap. I really think we could work together on a solution for this. Please let me know if you are willing to schedule a call. Others interested in potentially contributing to this project please contact me.