We will now conclude on the findings of this work, looking at the viablility of trade aggregation, but also of the scaling implications for other dapps or usecases.

0.1 zkSwap

In general, the approach explored in this work seems to be viable. We were able to build a trustless system, that successfully reduces the gas required for trading. The entire system is also non-custodial, mirroring the properties of decentralized exchanges. The data availability problem has also been solved, reaching the goals set for this work. The gas savings also look encouraging. We're able to reduce the cost per trade in this implementation, increasing the batch size can reduce the gas cost per trade even further. While the smart-contract logic used for verifying aggregation batches can still be optimized for gas efficiency, it could process much larger batch sizes, without many changes. It can not be considered a bottleneck in this system, and is only limited by the Ethereums block gas limit. The bottleneck of the system is the aggregator, and the complexity of computing the proofs for each batch. The approaches to scale the aggregation batches can broadly be separated into two groups.

Reducing Constraint Count Zk-rollup enabled applications largely rely on the hashing functions to function. For one, its thought to be the most efficient approach to store state in a merkle tree. A trees state can be commited on-chain cheaply by storing the root, and an inclusion proofs complexity correlates logarithmically to the trees depth. This however, requires a large amount of hashing in the circuits. New hashing functions like Poseidon look promising in reducing the constraint count per hashing operation, but the security of these functions is still being researched. Some hashes need to be computed in a circuit, but also on-chain. When computing a hash in a circuit and on-chain, the decision to opt for a hashing function that can be computed efficiently on-chain is obvious, after all we're optimizing for gas consumption. However, this increases our circuit size substantially, thereby reducing the batch sizes that can be aggregated in a reasonable amount of time. Having access to a hashing function that can efficiently be used in a circuit and on-chain would increase the throughput of zk-rollup enabled applications significantly.

maybe mention percent of constraints from hashing?

Increasing Constraint Throughput The throughput of an aggregation batch can also be increased by speeding up the proving steps. A large speed up could be achived by parallelizing the proving steps, which would make larger batch sizes viable. As shown by Loopring, there seem to be a number of areas where large efficency gains can be achived.

ZK-rollup as a Whole Considering that there is a clear path for improvement, zk-rollup is a strong contender to scale the Ethereum blockchain to process orders of magnitude more transactions. This work has also shown, that it is viable to aggregate user transactions for existing smart-contracts. Having to interact with other smart-contracts to execute an aggregation batch does add complexities and requires more gas to verify on-chain. The gas savings that can be achived still make it a viable approach. Another thing to consider when working with other smart-contracts in a zk-rollup application are the return values from those smart-contracts. In our case for example, the amount received when executing a Uniswap trade can't be predicted. While solving these issues add even more complexities and inefficiencies to the protocol, they can be solved with current technologies. However, it remains to be seen how much the technology continues to develop. The potential of recursive PLONK proofs cant be understated here. Being able to recursivly prove arbitrary circuits, resulting in only one on-chain verification transaction could enable scaling solutions, far superior to the approach presented here. Entire smart-contract ecosystems can potentially be brought to layer-2, Ethereums blockchain only being used to commit and emit state. Different zk-rollup systems can also interact with each other in a trustless and decentralized manner, ensuring composability among smart-contracts, even if deployed on different zk-rollup systems. The literatur on Zinc and the utilization of recursive PLONK is far from ideal, so it must be taken with a grain of salt. Given that Matterlabs, the company behind zkSync and Zinc, is experienced in this field and the potential implications of this technology, its worth mentioning at this point. A testnet utilizing early versions are also online, with an example app being deployed.