Duke CS+ Summer 2021: Decentralized Finance: Cryptocurrency and Blockchain on the Internet Computer: https://ic.pubpub.org/pub/7ms2ilwh/

Orthogonal Persistence as a Solution For Decentralized Data Storage

By Rhys Banerjee

Project Leads: Profs. Luyao Zhang, Yulin Liu, Kartik Nayak, Fan Zhang
Graduate mentor: Derrick Adam

Project Summary

The proposed research is aimed at experimental investigation of the differences in data efficiency between the Liquity Protocol (on Ethereum) and the WaterPark Protocol (on the Internet Computer). The main difference between the Internet Computer and Ethereum is the Internet Computer's use of orthogonal persistence. Thus, by measuring gas costs between an orthogonally persistent system, and one that lacks orthogonal persistence, we will be enabled to make inferences on the impacts that orthogonally persistent data makes with the Internet Computer. How do differences in persistence alter the storage efficiency of WaterPark as compared to Liquity? What role can orthogonal persistence play in financial applications? Does saved data due to orthogonal persistence have any tangible effect on the application itself?

Intellectual Merit

Duke CS+ Summer 2021: Decentralized Finance: Cryptocurrency and Blockchain on the Internet Computer: https://ic.pubpub.org/pub/7ms2ilwh/

The WaterPark protocol, if successful, will likely make a large contribution to FinTech, blockchain, and persistence. Orthogonal persistence is not a new concept, however, its application to a protocol on a blockchain is. The application of orthogonal persistence within the context of a more efficient decentralized borrowing protocol without a lender is novel. The current literature on orthogonal persistence consists of its applications to programming languages like Java, and criticisms of the concept itself. The literature on orthogonal persistence is sparse. The application of the practice in this new environment seeks to create a solution to the storage issues plagued by blockchain in the past, for a product with consumer merit as well.

Table of Contents

Abstract

Persistence refers to object and process characteristics that continue to exist even after the process that created it ceases or the machine it is running on is powered off. Persistence is said to

be "orthogonal" when implemented as an intrinsic property of the execution environment of a program. Orthogonal Persistence is not a new concept[1], nor one created by the Dfinity Foundation, however, it is one that aims to fundamentally alter both the efficiency of speed and scaling of modern computing. Moreover, while Liquity AG has developed an effective decentralized lending protocol on the Ethereum Network, this protocol could be improved upon by implementation on a different blockchain network. The network we have chosen to adapt Liquity on is Dfinity's Internet Computer, which serves as a more scalable and data-efficient alternative to the Ethereum Network.

For this study, we seek to understand orthogonal persistence as a solution to data storage. Having adapted the Liquity protocol to the Internet Computer in an application titled WaterPark, our explicit research question is: how do differences in persistence alter the storage efficiency of WaterPark as compared to Liquity? The methodology for experimentation will be conducted via user testing in a procedure in which we compare gas costs between Liquity on Ethereum and WaterPark on the Internet Computer as a way of determining differences in data efficiency. The merits of this experimentation include the advent of a data-efficient decentralized lending protocol -- one which lacks a lender. This is a truly innovative concept and one which can only be efficiently done on the Internet Computer.

Introduction and Background

The Internet Computer exists in a rapidly evolving sphere of Decentralized Finance, a growing field of economics that poses to change the very way we think about money. The world's most popular DeFi system is undoubtedly the Ethereum Network. Launched in 2015, the influence of Ethereum is akin to a stepping stone in the world of decentralized finance and

[1] Alan Dearle, Graham N.C. Kirby and Ron Morrison: "Orthogonal Persistence Revisited"

cryptocurrency. It must be said, however, that Ethereum has some scaling issues that prevent it from being the DeFi platform of choice for a scaling decentralized borrowing protocol.

To store 1GB of data inside an Ethereum smart contract would cost millions of dollars, which can make it far too expensive to maintain anything beyond fiduciary data.[2] This makes it difficult to rely on Ethereum for complex protocols meant to scale.

This can be contrasted with the Internet Computer. Launched in 2020 by the Dfinity Foundation, the Internet Computer acts as a counterpart to Ethereum, improving on it especially in the metric of scalability and gas costs.[3] The Internet Computer is more efficient than Ethereum for a myriad of reasons, however, one of the reasons -- which will be the focus of this paper-- is its use of orthogonal persistence in data storage. Persistence refers to object and process characteristics that continue to exist even after the process that created it ceases or the machine it is running on is powered off.[4] Persistence is said to be "orthogonal" when it is implemented as an intrinsic property of the execution environment of a program. Orthogonal persistence can act as a significant boon for specific protocols which already exist on Ethereum due to promises of better scalability. Case-in-point, we focus our attention on Ethereum's Liquity protocol.

Liquity

Liquity is an already existing protocol on the Ethereum Network. We seek to summarize the main procedure of the protocol. Users are able to open an account, called a trove, and put down ETH as collateral for under-collateralized loans and receive back an interest-free loan in

[2] Williams, D. (2021, June 6). How Ethereum Could Be Supercharged by the Internet Computer Network. Medium.https://medium.com/dfinity/how-ethereum-could-be-supercharged-by-the-internet-computer-network-afc513bf15e1.
[3] Ibid.
[4] Editors of CLiki. (n.d.). Orthogonal Persistence. CTO : Orthogonal Persistence. http://tunes.org/wiki/orthogonal_20persistence.html.

LUSD (a stablecoin pegged to USD). This is in effect so long as the user maintains a collateral ratio of 110%. If a user's collateral ratio falls below this amount, then their trove becomes liquidated, with their LUSD burned and their ETH being retrieved by the system.[5] Something novel about this is the lack of a lender and the automated nature of the whole system. In addition, there is also the presence of a mechanic entitled the 'stability pool.' Funds from the stability pool are used to ensure that the system always has enough solvency to pay for liquidated troves. When any trove is liquidated, an amount of LUSD corresponding to the remaining debt of the trove is burned from the Stability Pool's balance to repay its debt. In exchange, the entire collateral from the trove is transferred to the stability pool.[6] Funds comprising the supply are added by users, who are rewarded for depositing LUSD into the stability pool. One way they are rewarded is a guaranteed proportional share of ETH from liquidated troves. Another way is the regular allocation of a reward token -- LQTY -- to users who keep their LUSD in the stability pool. Overall, users who contribute to the Stability pool, called Stability Providers, are compensated for their services.

That is not to say that Liquity runs without flaws on Ethereum. Scaling issues with Ethereum heavily impact how users interact with the stability pool. The fact that liquidated troves decrease all deposits in the Stability Pool leads to an implementation challenge: due to Ethereum's block gas limit, it becomes impossible to iterate over all deposits in the stability pool for a large number of deposits. This has complexity O(N) and doesn't scale efficiently.[7] Due to inherent issues with Ethereum, finding a solution within the network is ill-advised. This is why we seek to solve this issue utilizing the Internet Computer.

---

[5] Lauko, R., & Pardoe, R. (2021, April 5). Liquity: Decentralized Borrowing Protocol.
[6] Ibid
[7] Pardoe, R. (2021, January 21). Scaling Liquity's Stability Pool. Medium. https://medium.com/liquity/scaling-liquitys-stability-pool-c4c6572cf275.

When building on the Internet Computer, we notice it can be characterized by its low gas costs: whereas storing 1GB of data in an Ethereum smart contract would realistically cost millions of dollars, the same feat could be achieved on the Internet Computer for as little as a few cents.[8] It can also be characterized by its high speeds: the Internet Computer enables decentralized services at speeds familiar to internet users, by running canisters on made-for-purpose machines in data centers around the world.[9]

When considering the benefits of adapting the Liquity protocol for the Internet Computer, we must observe the very nature of how the entire network was built to scale. The very foundations of the network were created with scalability and efficiency in mind. The best example of this is the integration of orthogonal persistence in the native programming language of the Internet Computer, Motoko.

Motoko

When developing WaterPark, we have been utilizing Motoko for back-end development. Motoko is the native programming language on the Internet Computer. It runs on WebAssembly (Wasm) and it was built with orthogonal persistence in mind. A consideration that the Motoko team maintained was the idea of allowing developers to use blockchain technology without first learning a different form of computing.[10] What Motoko lacks (for blockchain developers): no observable notion of block or block height, no explicit constructs for updating state on the blockchain, no other API for writing data to persistent storage, like files or databases.[11] Instead,

---

[8] Williams, Dominic: “How Ethereum Could Be Supercharged by the Internet Computer Network”
[9] Ibid
[10] Rossberg, A. (2021, June 10). Motoko, a Programming Language Designed for the Internet Computer, Is Now Open Source.Medium.https://medium.com/dfinity/motoko-a-programming-language-designed-for-the-internet-computer-is-now-open-source-8d85da4db735.
[11] Ibid

this goes back to orthogonal persistence. In essence, this means that developers do not have to worry about explicitly saving their data between messages or bother with an external database. Whatever values or data structures are stored in program variables will still be there when the next message arrives.[12] Since data is saved, developers do not have to bother with dedicated space to saving their data manually, saving not only their own time but also storage.

This is possible because of Motoko's use of WebAssembly (Wasm). Since the state of a Wasm module is clearly isolated in a module's memory, globals, and tables, this allows for the platform to take care of transparently saving and restoring the private state of a canister between method invocations.[13] Canisters can only hold up to 4GB of memory pages due to the limitations of WebAssembly implementations, however, this can be expanded through the use of inter-canister calls.[14] A system can be composed of many canisters interacting so that there is no upper limit on the amount of memory a system can hold. Motoko users can import "BigMap" for this purpose.

BigMap is an upcoming Motoko plugin created by Dfinity to provide a library for building apps that scale without bound. By using the BigMap library as a back-end service, applications can dynamically chunk, serialize, and distribute data to multiple canisters.[15] What purpose does BigMap serve, though? Since the Internet Computer protocol relies on replicated state across nodes in a subnet, it can provide certain guarantees regarding fault tolerance and failover natively that are not generally available to applications running on other platforms or protocols. So, since the Internet Computer can ensure that canisters are available to receive and

---

[12] Ibid
[13] Ibid
[14] DFINITY. (n.d.). Internet Computer: Documentation. Create scalable apps. https://sdk.dfinity.org/docs/developers-guide/tutorials/scalability-cancan.html.
[15] Ibid

respond to requests, the hash table back-end service was replaced with a scalable backend called BigMap.

BigMap offers building blocks for application-specific, in-memory data abstractions that scale using any number of canisters. In the case of the Dfinity-created application, CanCan, each canister still has limited capacity, but the application uses the canisters it needs and keeps track of the fragments that make up the full video content for each user's videos in an index file called the manifest.[16]

Tying it back to WaterPark

As stated earlier, my partner Oum Lahade and I are working on adapting the Liquity protocol for the Internet Computer into an application titled WaterPark. Our protocol carries over certain fundamental components of the Liquity architecture. Certain canisters are based specifically on elements of Liquity: for instance, the user canister, and the stability canister are based entirely on Liquity users, and the stability pool respectively. The WaterPark will also replicate the LUSD stablecoin's general stability mechanism. Figure 1 is a diagram created by Oum Lahade meant to illustrate how the different canisters are meant to interact with each other during a standard WaterPark protocol:

[16] Ibid

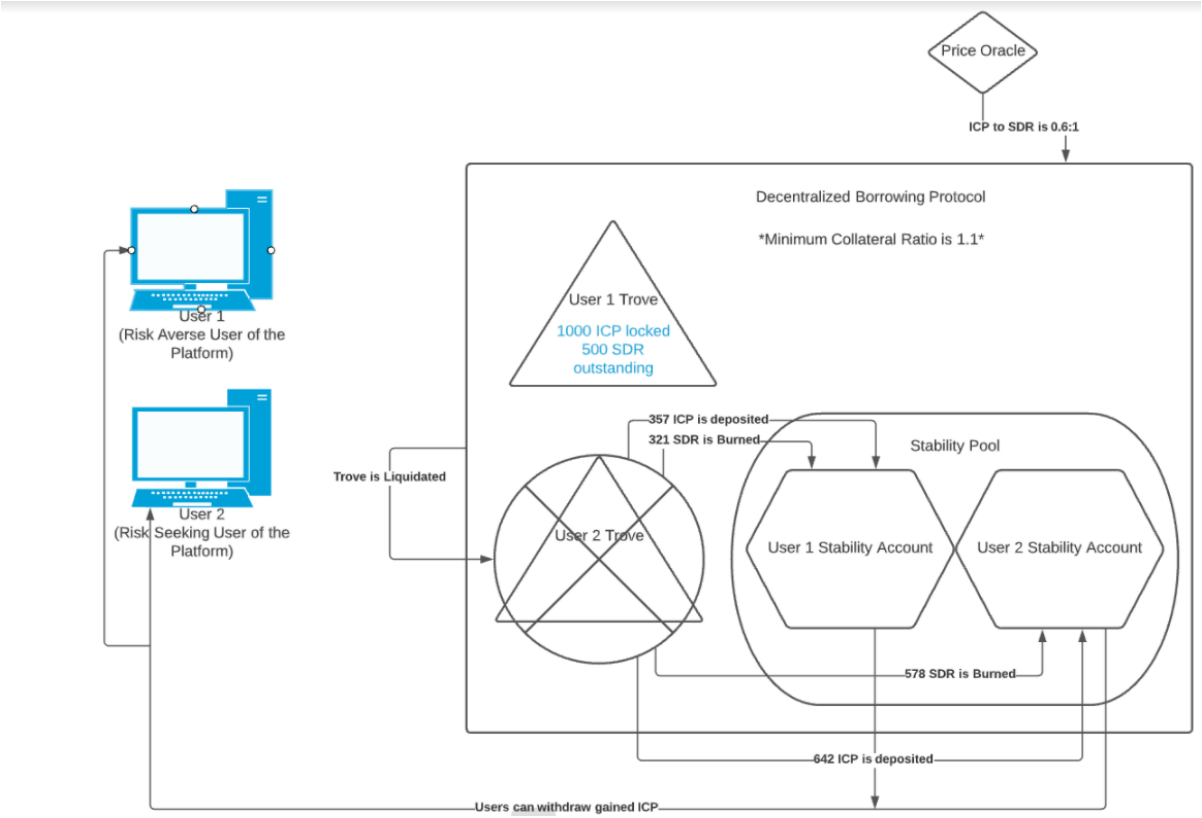

Figure 1: As User 2's collateral ratio falls underneath 110%, their trove becomes liquidated. Illustrated by Oum Lahade.

There are certain places where WaterPark differs from Liquity. Besides using ICP in place of ETH and our ERC-20 style SDR-pegged stablecoin in place of LUSD, we lack rewards token in place of LQTY. Additionally, WaterPark uses trove-like stability accounts to distribute rewards for users that deposit to the stability pool. Moreover, WaterPark will store all data on-chain on the Internet Computer network. Finally, while Liquity lacks a front-end developed by Liquity AG, we are currently in the development of a front-end.

Research Question and Methodology

We wish to look at orthogonal persistence as a solution to data storage. To this end, our main research question asks us to compare Liquity and WaterPark, thus giving us insight into the

differences between the Internet Computer and Ethereum as a whole. My research question asks explicitly: “how do differences in persistence alter the storage efficiency of WaterPark as compared to Liquity?” We wish to analyze orthogonal persistence as a solution to data storage, and so our methodology will consist of us replicating a version of WaterPark without orthogonal persistence and comparing gas costs between this version and WaterPark. We will use these gas costs as a proxy for analyzing data efficiency, assuming that higher gas costs correlate to lower efficiency.

Novelty and Contribution to Literature

The WaterPark protocol, if successful, will likely make a large contribution to FinTech, blockchain, and persistence. As stated earlier, orthogonal persistence is not a new concept, however, its application to a protocol on a blockchain is. The application of orthogonal persistence within the context of a more efficient decentralized borrowing protocol without a lender is novel.

The current literature on orthogonal persistence consists of its applications to programming languages like Java, and criticisms of the concept itself. The literature on orthogonal persistence is sparse. The application of the practice in this new environment seeks to create a solution to the storage issues plagued by blockchain in the past, for a product with consumer merit as well.

Bibliography

Alan Dearle, Graham N.C. Kirby and Ron Morrison: “Orthogonal Persistence Revisited”

Duke CS+ Summer 2021: Decentralized Finance: Cryptocurrency and Blockchain on the Internet Computer: https://ic.pubpub.org/pub/7ms2ilwh/

DFINITY. (n.d.). *Internet Computer: Documentation*. Create scalable apps. https://sdk.dfinity.org/docs/developers-guide/tutorials/scalability-cancan.html.

Editors of CLiki. (n.d.). *Orthogonal Persistence*. CTO : Orthogonal Persistence. http://tunes.org/wiki/orthogonal_20persistence.html.

Lauko, R., & Pardoe, R. (2021, April 5). Liquity: Decentralized Borrowing Protocol.

Pardoe, R. (2021, January 21). *Scaling Liquity's Stability Pool*. Medium. https://medium.com/liquity/scaling-liquitys-stability-pool-c4c6572cf275.

Rossberg, A. (2021, June 10). *Motoko, a Programming Language Designed for the Internet Computer, Is Now Open Source*.Medium.https://medium.com/dfinity/motoko-a-programming-language-designed-for-the-internet-computer-is-now-open-source-8d85da4db735.

Williams, D. (2021, June 6). *How Ethereum Could Be Supercharged by the Internet Computer Network*. Medium.https://medium.com/dfinity/how-ethereum-could-be-supercharged-by-the-internet-computer-network-afc513bf15e1.