Assessing Blockchain Protocols in Reputation-Based Networks

Sarayu Namineni, advised by Prof. Seth Goldstein

Carnegie Mellon University, Computer Science Department

Abstract/overview

Building on Local Trust (BoLT) is a reputation-based lending platform that uses a public ledger to create a transparent and universally accessible system. Our project aims to improve the performance of the BoLT ledger by integrating layer-1 and layer-2 blockchain protocols into our application. In this proposal, we assess the flexibility and performance of a highly-scalable layer-1 solution, Solana, in the context of the BoLT ecosystem.

Introduction/motivation

In this project, we implement a prototype of the BoLT ledger on top of the Solana blockchain in order to assess the flexibility and performance of its smart contract language. Solana is a high-performance layer-1 blockchain protocol which uses a hybrid of Proof-of-Stake (PoS) and Proof-of-History (PoH) to reach consensus. It boasts a throughput with a theoretical upper limit equivalent to centralized databases and in practice, its throughput is higher and its costs are lower than its main competitors.

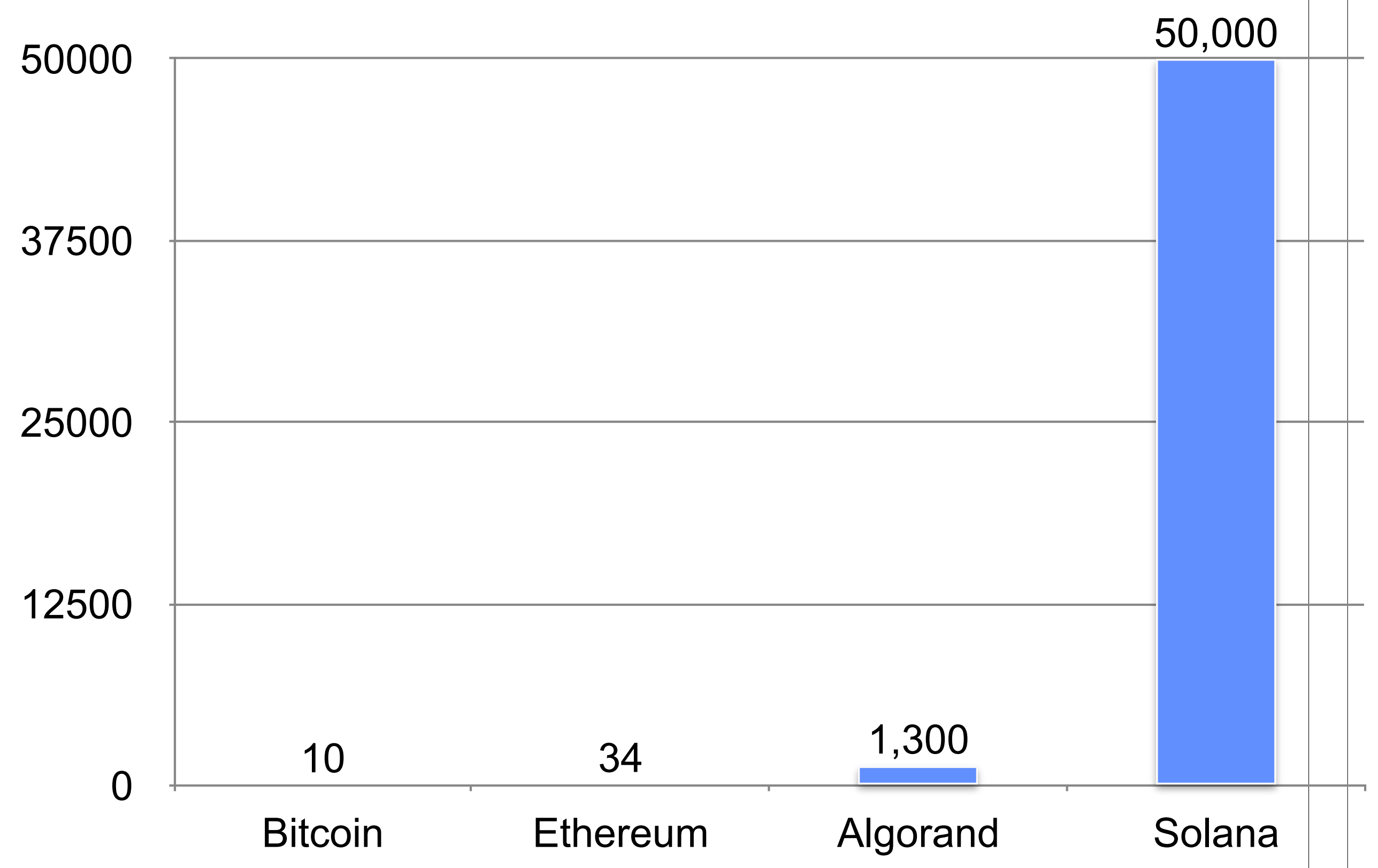


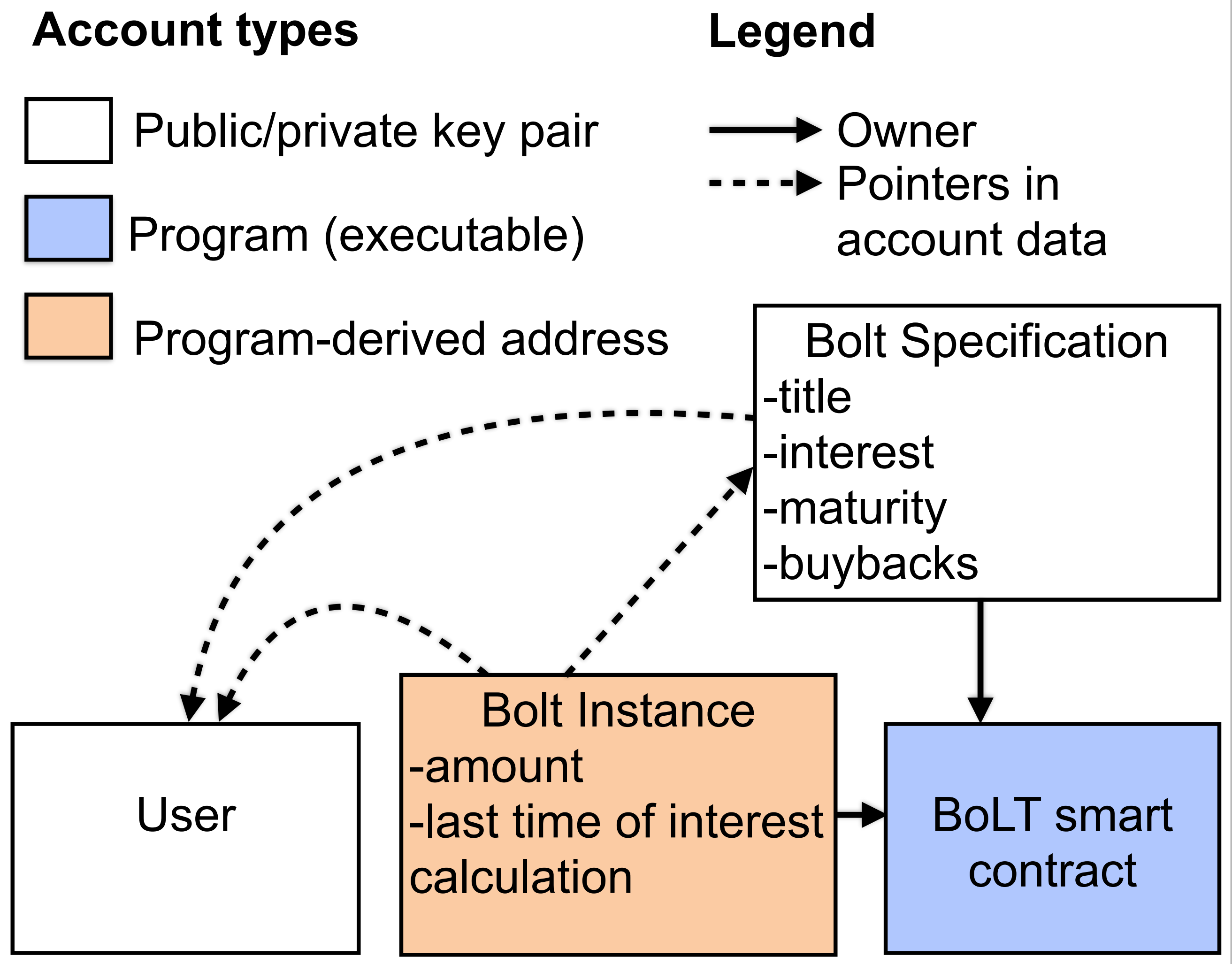
Fig. 1: Throughput of mainstream blockchains

Solana outperforms the most popular blockchains, Bitcoin and Ethereum, as well as newer blockchain technologies that rely on PoS consensus, such as Algorand (Fig. 1).

Solana's cost model also differs from its competitors in significant ways. Most blockchains have "gas fees," or costs for performing queries or transactions on the blockchain. For instance, Ethereum's gas fees range between $5 and $150. Instead of having a gas fee, Solana charges all accounts with a rent fee, or a cost associated with storing the data on the blockchain. It is common for users to make their accounts "rent-exempt," or to pay for 2 years of rent at a time. This means that Solana's cost model is largely proportional to the number of bytes of account data that are stored per smart contract.

PROPOSED METHODS/DESCRIPTION

The architecture for the ledger prototype is largely based on the Solana Program Library (SPL) token program (Fig. 2). The BoLT smart contract program differs from SPL in the fact that it allows users to define variable length specifications with multiple interest rates, maturity rates, and buyback options. This prototype of the BoLT ledger supports instructions such as defining bolt specifications and minting and transferring bolt instances.



Account types

□ Public/private key pair

🟦 Program (executable)

🟧 Program-derived address

Legend

→ Owner

⇢ Pointers in account data

Bolt Specification
-title
-interest
-maturity
-buybacks

User

Bolt Instance
-amount
-last time of interest calculation

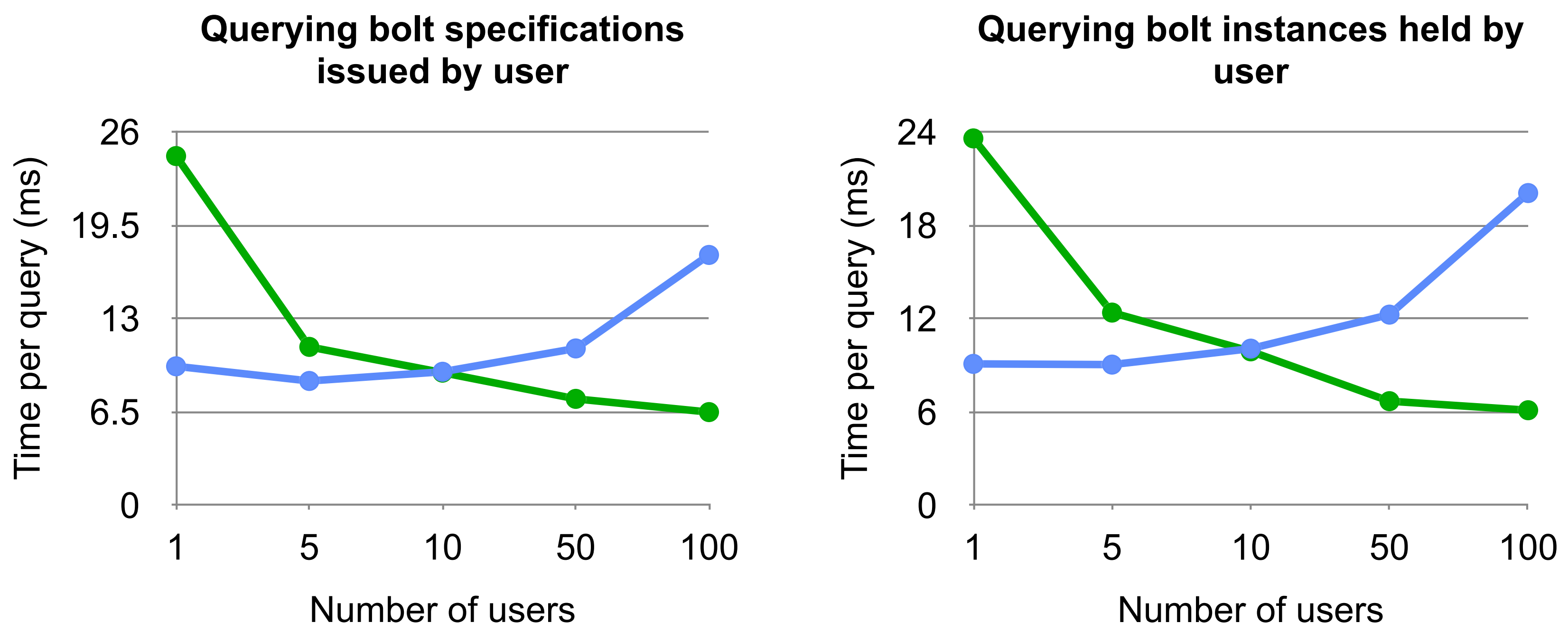
BoLT smart contract

Fig. 2: Architecture of BoLT smart contract

All data, including the executable smart contract that represents the ledger itself, is stored in its own account on the Solana blockchain. Accounts are addressable by a key, which can be a public key or a program-derived address, the main difference being that a program-derived address does not have an associated private key. Accounts are owned by programs, or executable accounts. Only the owner of an account is allowed to modify its data.

RESULTS

We measure the difference in performance and cost when using 4-byte unique identifiers instead of 32-byte public keys to address accounts. In all tests, there are 1000 bolt instances, 1000 bolt specifications, and 100 users.



Task	4-byte UID	32-byte public keys
Querying specs	4.024	0.069
Querying instances (filtration)	5.444	17.231
Querying instances (computation)	10.945	0.497
Querying specs per user	6.479	17.422
Querying instances per user	6.107	20.066
Minting	448.342	437.060
Transferring	453.908	444.833

Fig. 3: Summary table of times per task

CONCLUSIONS

Having 4-byte unique identifiers improves performance as well as costs. This is because the main method for querying is to filter through all the accounts owned by our smart contract and compare bytes of account data at specified offsets. With smaller identifiers, we have less data to compare, so as the number of accounts we need to query increases, we begin to see improvements in query performance. Furthermore, since costs are proportional to the amount of data stored, addressing accounts by 4-bytes instead of 32-bytes results in a reduction in storage, and thereby, costs.