Chainspace: A Sharded Smart Contracts Platform

Written By:

Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis

Presented By:

Aaron Zhang

Outline

- Introduction
- System Overview
- The Chainspace Application Interface
- The Chainspace System Design
- Implementation and Evaluation
- Limitations
- Conclusions

Outline

- Introduction
- System Overview
- The Chainspace Application Interface
- The Chainspace System Design
- Implementation and Evaluation
- Limitations
- Conclusions

Modern block chains are very slow when significant volume is applied.

Chainspace

- Chainspace is a distributed ledger platform for high-integrity and transparent processing of transactions within a decentralized system.
- It avoids the high latency that Ethereum faced by sharding the coin base.
- Supports a new form of privacy by separating the code that executes commands and those that check the computation.

Outline

- Introduction
- System Overview
- The Chainspace Application Interface
- The Chainspace System Design
- Implementation and Evaluation
- Limitations
- Conclusions

Objects

- Items that hold state in Chainspace.
- Refer to the objects as o and a set of objects as $o \in O$.
- Objects have an ID and a Type.
- Objects are either active or inactive.

Contracts

- Special type of object that has functions and data regarding itself.
- A namespace for Objects within Chainspace.
- Refer to contracts as c

Procedures

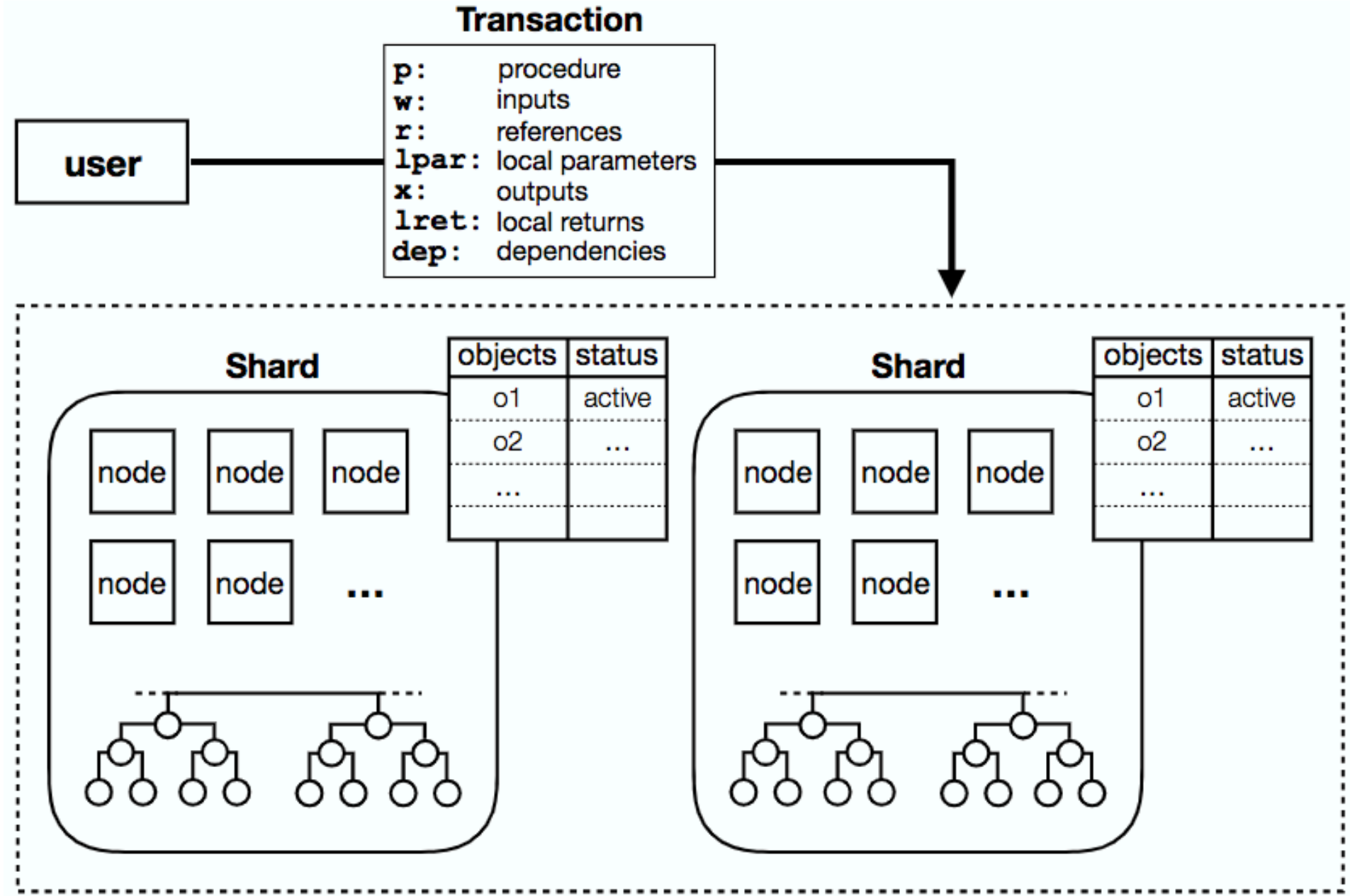
- Where a number of objects are processed to generate some output of objects.

- c. p(~w,~r, lpar, spar)→ ~x, lret,sret

Checkers

- Every contract has a checker which recieves a procedure and then checks if the procedure is valid.

- c.v(p, ~w, ~r, lpar, ~x, lret, dep)→{ true,false }


Transaction
p: procedure
w: inputs
r: references
lpar: local parameters
x: outputs
lret: local returns
dep: dependencies
user
Shard
objects
status
o1
active
o2
...
node
node
node
node
node
...
Shard
objects
status
o1
active
o2
...
node
node
node
node
node
...

- Honest Shards
  - A shard where less than 1/3rd of all nodes are not controlled by an attacker.

- Dishonest Shards
  - A shard where more than 1/3rd or all nodes are controlled by an attacker.

Chainspace supports Security Properties

- Transparency
  - Every node in the Chainspace publishes a Merkle Tree type object which provides a history of changes to that node itself.
- Integrity
  - No two nodes in the shard can have the same data.
- Encapsulation
  - Shards cannot communicate with eachother.
- Non-repudiation
  - If a failure would allow those to add invalid transactions, using the Merkle Tree data, one can find where that data was added and take action.

Outline

- Introduction
- System Overview
- The Chainspace Application Interface
- The Chainspace System Design
- Implementation and Evaluation
- Limitations
- Conclusions

- Transactions are all compiled in a sequence of traces of the procedures that have been used to create them.

Two Rules to Modify Nodes within Contracts

- Sequence Rule
  - The trace list is only valid if its objects are in sequence
- Check Rule
  - Only transactions that pass the checker for the contract may be added.

Outline

- Introduction
- System Overview
- The Chainspace Application Interface
- The Chainspace System Design
- Implementation and Evaluation
- Limitations
- Conclusions

Directed Acrylic Graph

- A combination of objects, procedures, and outputs form a Directed Acrylic Graph which can be used to speed up a blockchain’s process.

- A unique identifier is attributed to each and every trace in the objects history for quick indexing.

Security Theorem 1

No sequence of valid transactions, by a polynomial time constrained adversary, may re-create an object with the same identifier with an object that has already been active in the system.

Proof

For two objects to have the same ID, they must have be within the same shard, be created by the same procedure, and have the same inputs.

When an object is created with at least one input, the active input objects are removed from the set and new nodes created with the same input object cannot be added.

Audits

- Partial Audit
  - A partial audit is simply when the blockchain returns a reply whether or not the transaction has been passed
- Full Audit
  - A full audit involves replaying all transactions from the beginning of time so the user can understand the traces that led to the current transaction.

Security Theorem 2

If a contract c appears in any trace within a transaction T, then the concerned nodes set Φ(T)will contain nodes in a shard managing an object O of a type from contract c

Proof

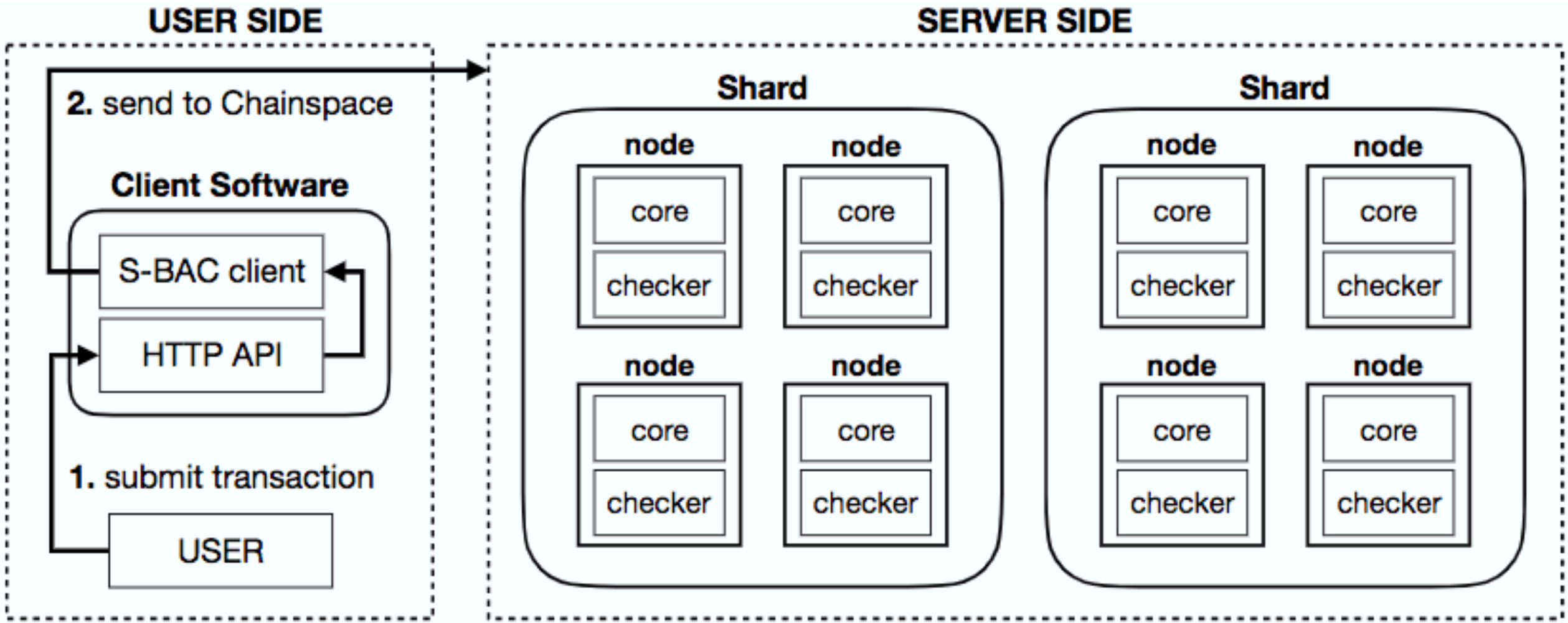
Transactions T cannot be placed within an object O without passing the checker function which only works if the object matches with the shard itself.

S-BAC

- Sharded Byzantine Atomic Commit
  - Combination of the Byzantine Agreement and an Atomic Commit
  - Byzantine Agreement
    - ensures that all honest members of a node side of 3f+1 will come to an agreement despite dishonest nodes of size smaller than f.
  - Atomic Commit
    - If one shard denies the commit, all shards deny the commit.

Outline

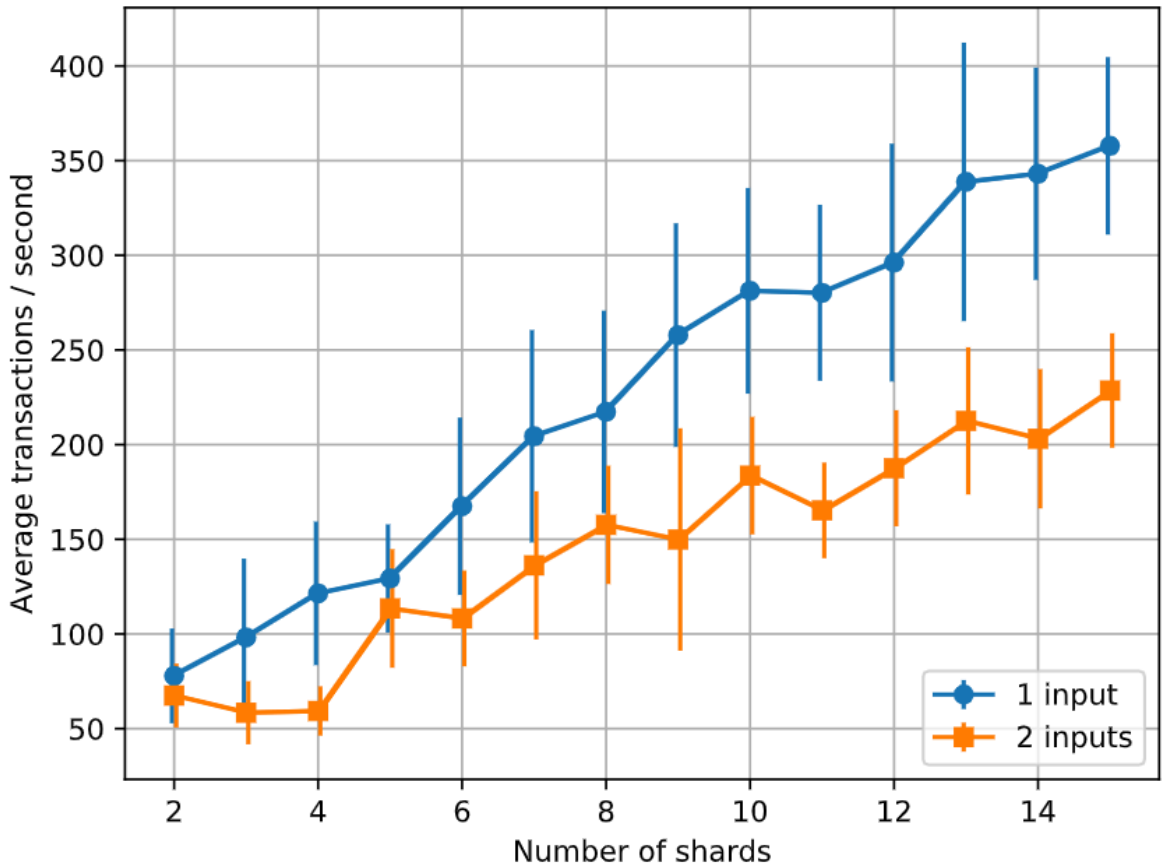
- Introduction
- System Overview
- The Chainspace Application Interface
- The Chainspace System Design
- System and Applications Smart Contracts
- Implementation and Evaluation
- Limitations
- Conclusions

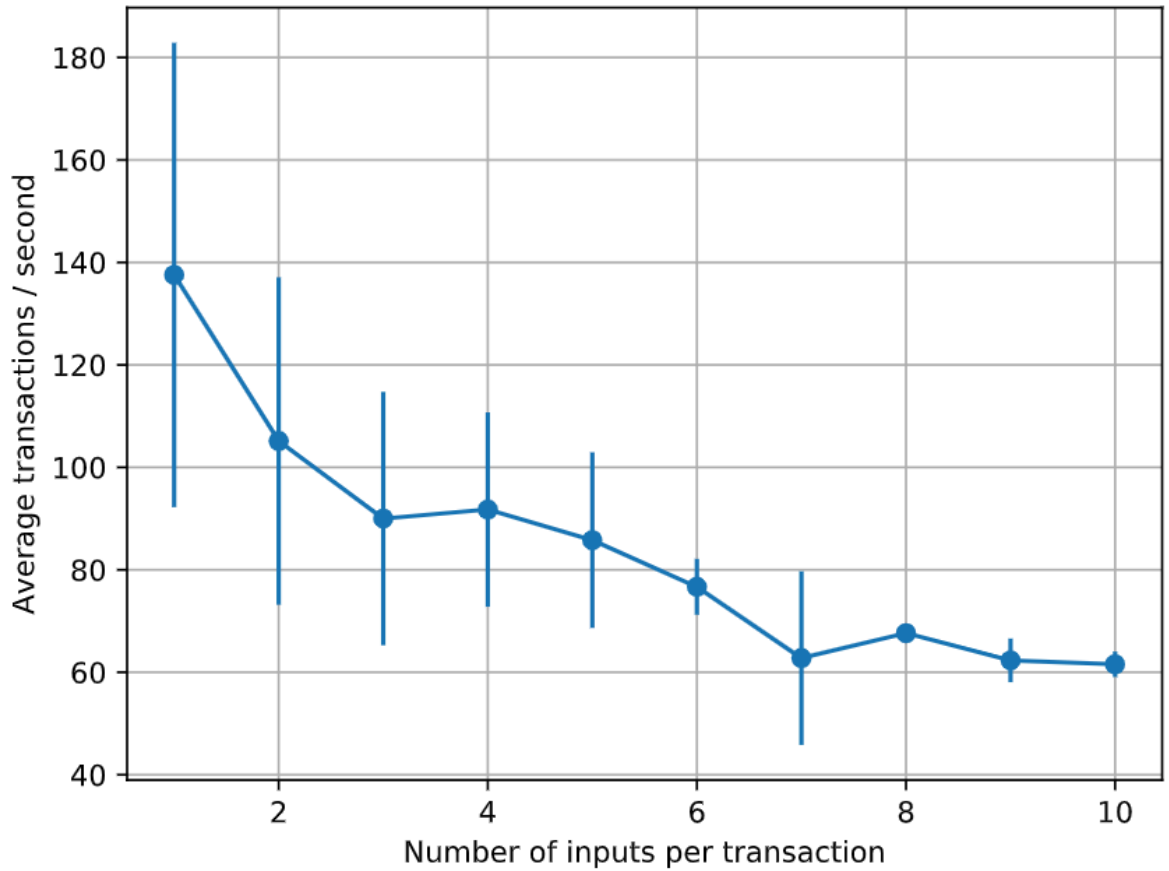

USER SIDE
SERVER SIDE
2. send to Chainspace
Client Software
S-BAC client
HTTP API
1. submit transaction
USER
Shard
Shard
node
core
checker

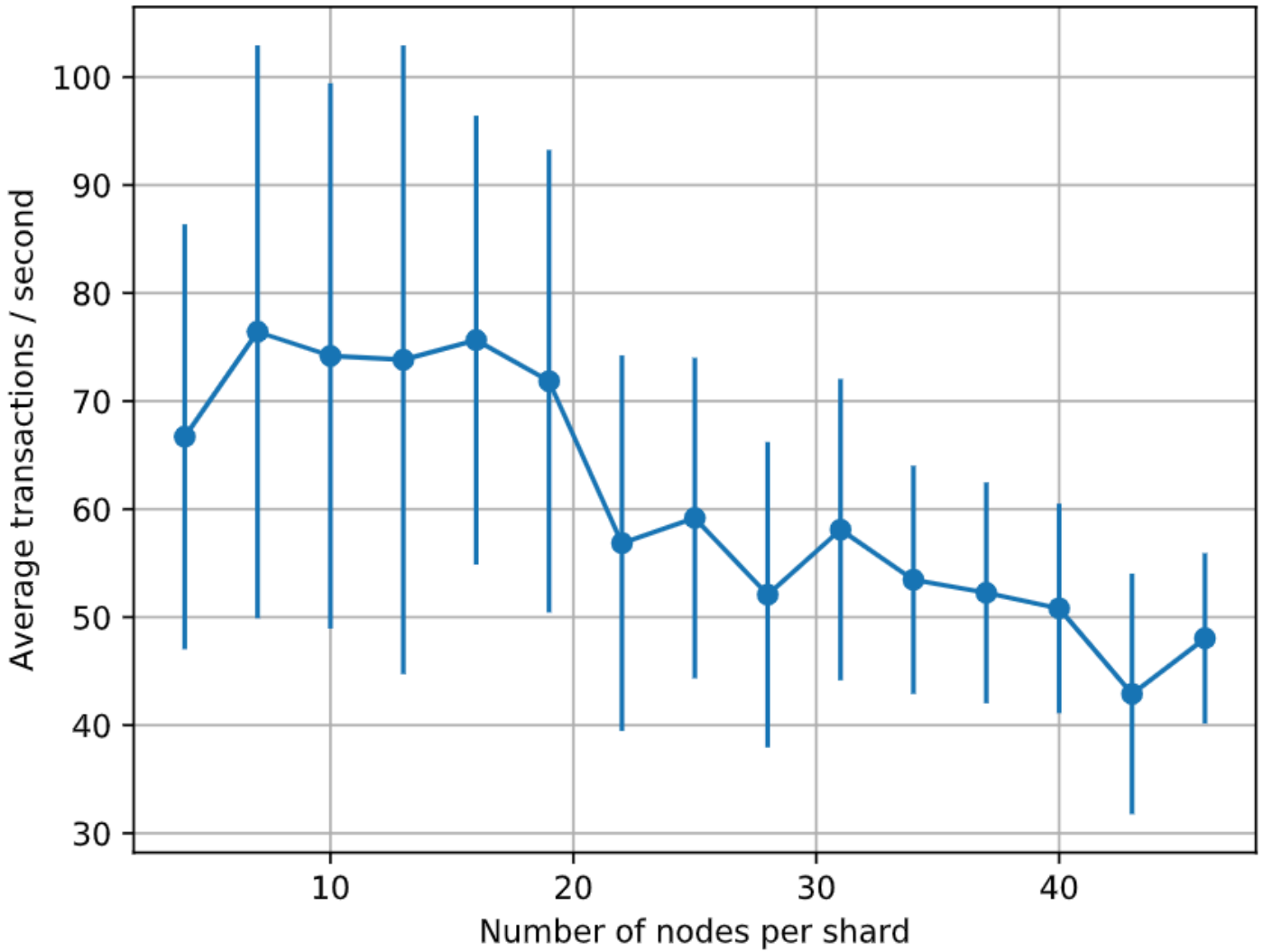
Implementation

- Implementation in Java(Nodes) and Python(Contracts)

- A node is made up of two parts
  - Checker, checks to see if the procedure matches with the node
  - Core, which can access the traces to find previous iterations of the node to compare.


Average transactions / second
400
350
300
250
200
150
100
50
2
4
6
8
10
12
14
Number of shards
1 input
2 inputs


Average transactions / second
180
160
140
120
100
80
60
40
2
4
6
8
10
Number of inputs per transaction


Average transactions / second
100
90
80
70
60
50
40
30
10
20
30
40
Number of nodes per shard

Outline

- Introduction
- System Overview
- The Chainspace Application Interface
- The Chainspace System Design
- System and Applications Smart Contracts
- Implementation and Evaluation
- Limitations
- Conclusions

Limitations

- A shard can be taken over if more than 1/3rd of its nodes are controlled by an attacker.
- Nodes from other shards can detect malicious shards but take no action.
- Checkers for each node is very costly in time.

Outline

- Introduction
- System Overview
- The Chainspace Application Interface
- The Chainspace System Design
- System and Applications Smart Contracts
- Implementation and Evaluation
- Limitations
- Conclusions

Conclusion

- Chainspace is a novel idea for expediting the transactions for a cryptocurrency that has a large load while increasing privacy.
- S-BAC is a new way that cryptocurrencies can authenticate themselves.

Questions?