Microsoft

Research Faculty Summit 2018

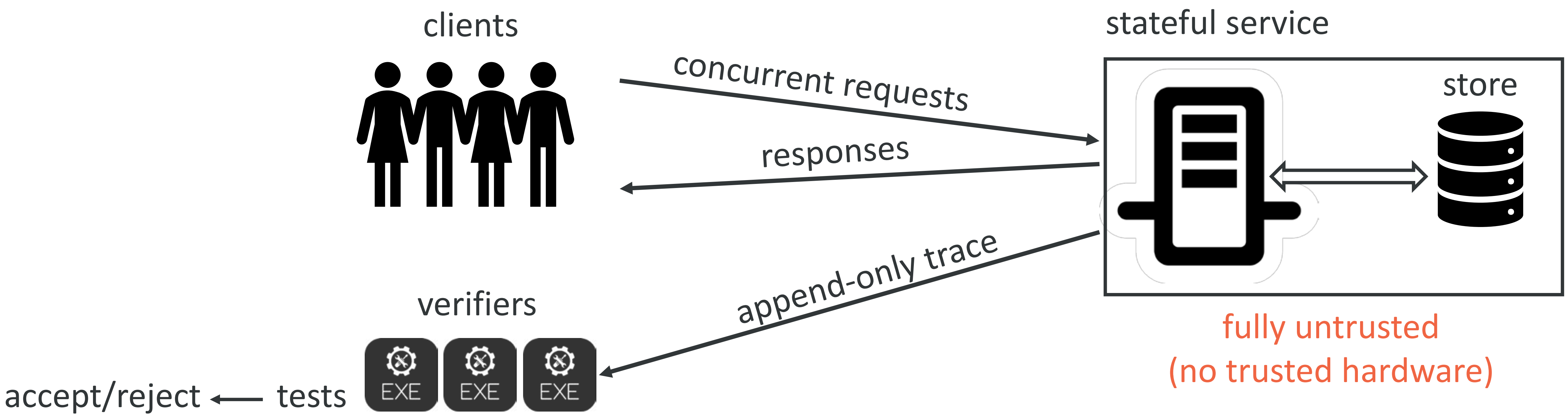
Systems | Fueling future disruptions

Spice: Verifiable state machines

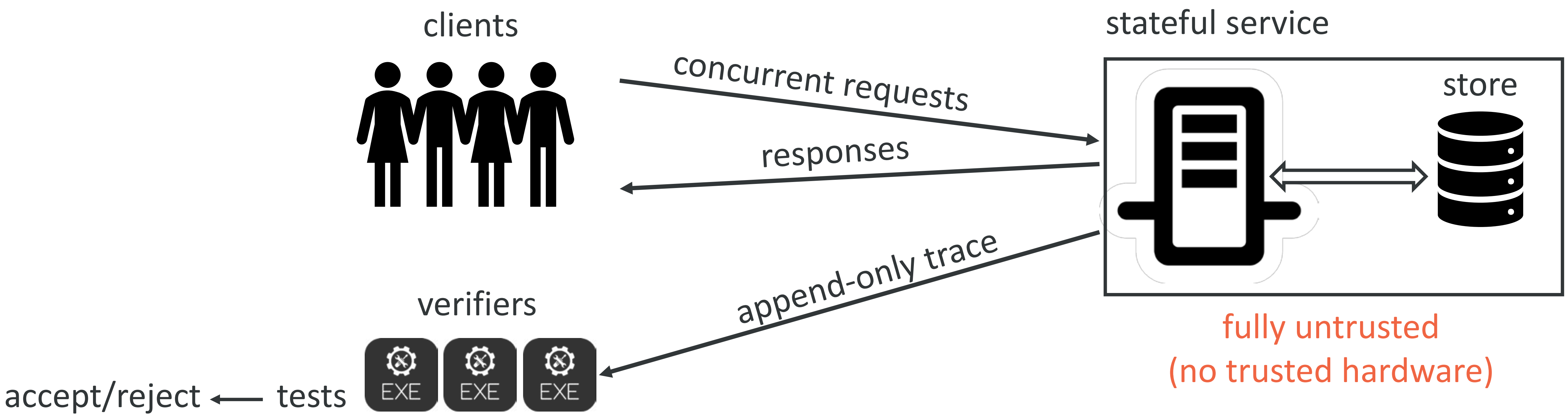
A foundation for building high-throughput confidential blockchains

Srinath Setty, Sebastian Angel, Trinabh Gupta, and Jonathan Lee
Microsoft Research UT Austin and NYU

Verifiable state machines



Verifiable state machines



- Correctness: If the service's behavior is equivalent to a serial execution, verifiers `accept`
- Soundness: If the service misbehaves, Pr[a verifier outputs `accept`] $< \epsilon$
- Zero-knowledge: Trace does not reveal *anything* beyond correct execution
- Succinctness: Each entry in the trace is small

Prior work on verifiable state machines

- The underlying theory dates back to 90s: Babai et al.[STOC91]

Cost reductions by 10^{20}x

- Pepper [HotOS11, NDSS12], CMT [ITCS12], Ginger [Security12], TRMP [HotCloud12]
- Zaatar[EuroSys13], Pinocchio[Oakland13], SNARKS-for-C[CRYPTO13]

Support stateful computations

- Pantry[SOSP13], Geppetto[Oakland15], CTV[EUROCRYPT15], vSQL[Oakland17], ...

Storage interfaces: key-value stores, SQL databases, etc.

Two key limitations of recent systems:

- Storage ops. are expensive: tens of seconds to minutes of CPU-time
- Support only a sequential execution model

Prior work can only support < 0.15 reqs/sec (even for simple services)

Support stateful computations

- Pantry[SOSP13], Geppetto[Oakland15], CTV[EUROCRYPT15], vSQL[Oakland17], …

Storage interfaces: key-value stores, SQL databases, etc.

Two key limitations of recent systems:

- Storage ops. are expensive: tens of seconds to minutes of CPU-time
- Support only a sequential execution model

Prior work can only support < 0.15 reqs/sec (even for simple services)

Our system, Spice [OSDI18, to appear]:

- Features a storage primitive that is >100x faster
- Supports a concurrent execution model
- Throughput: 488—1048 reqs/sec (512 CPU-cores)

Rest of this talk

- **Applications of verifiable state machines**
- Background and overview of Spice
- Experimental results

Why are we interested in verifiable state machines?

They enable us to build:

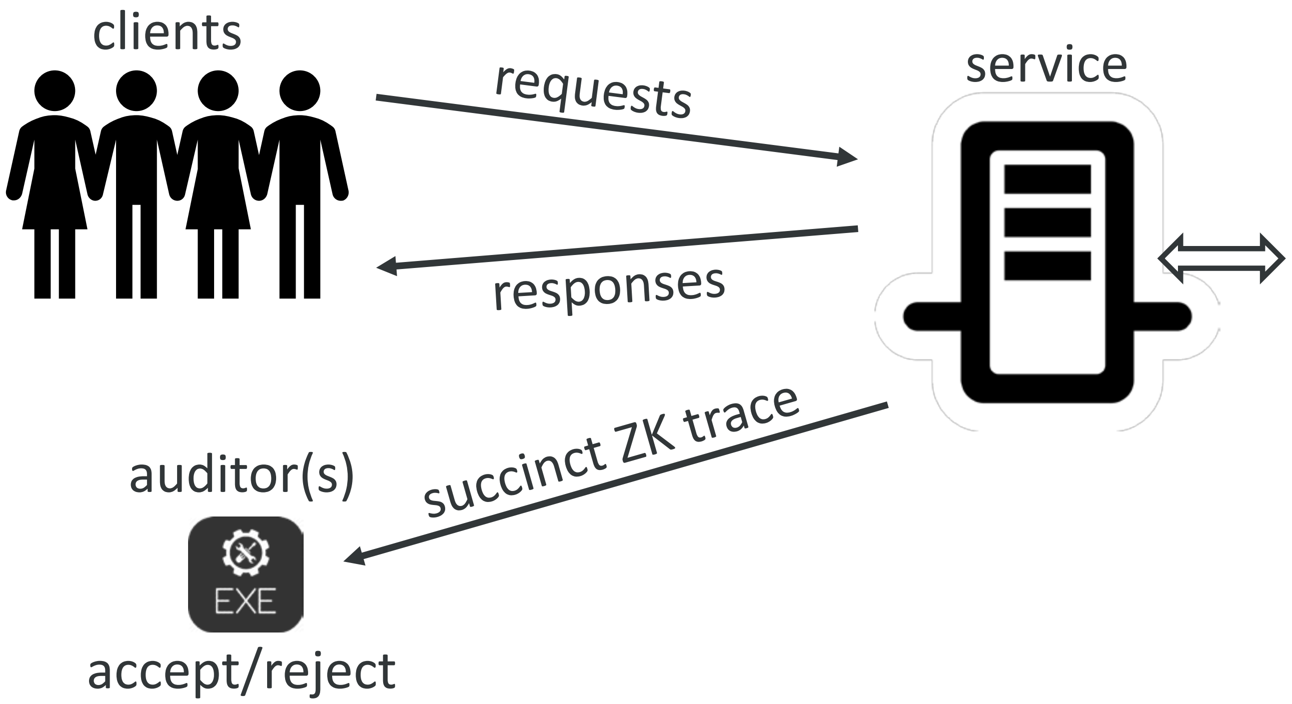
1. Cloud services *without trusting* the cloud infrastructure

2. *Private and efficient* blockchains
    - Permissionless (e.g., Ethereum)
    - Permissioned (e.g., Hyperledger Fabric, Quorum, etc.)

Cloud-hosted ledgers

(inspired by https://sequence.io)

WeChat Pay Square



state

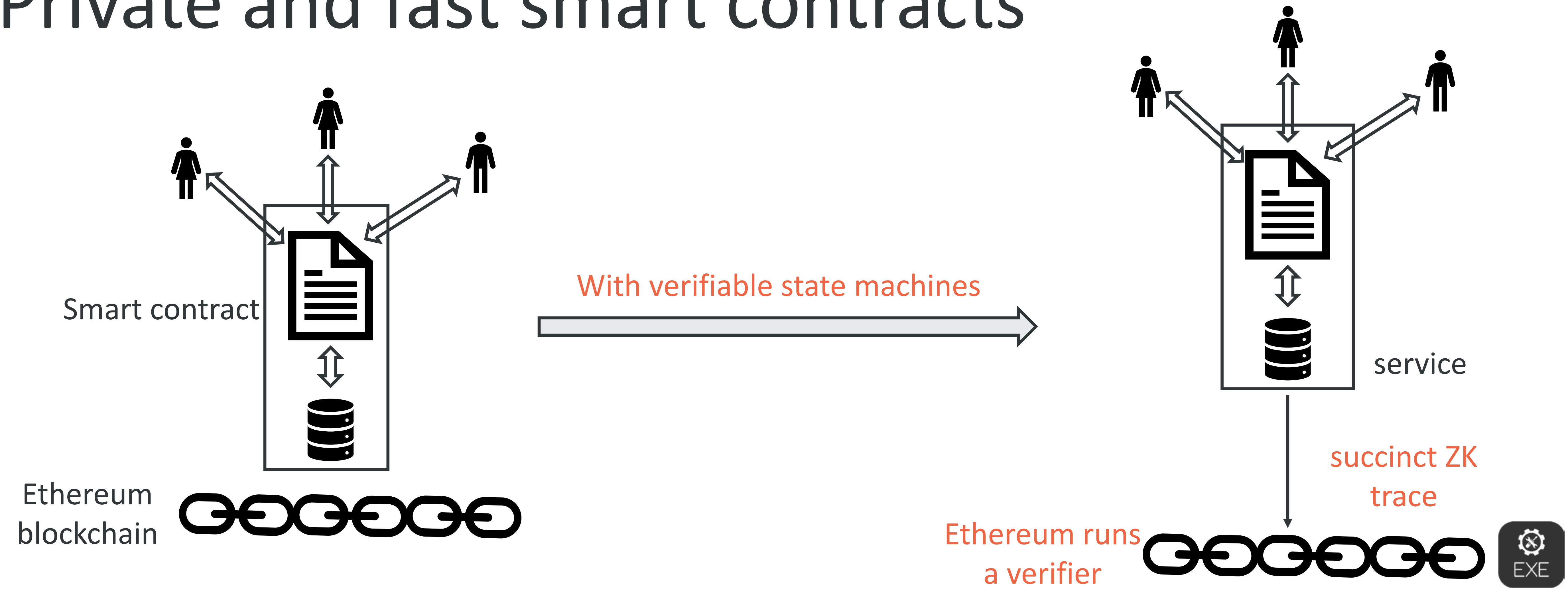
ClientId	Asset	Balance
0x2ff1	USD	200
0xfad1	USD	500
0xba21	Euro	100

Value proposition: An auditor can verify the service---without access to requests or trusting the service

1. `Issue(clientId, asset, balance, issuerSig)`
2. `Transfer(senderId, recvId, asset, amount, senderSig)`
3. `Retire(clientId, asset, amount, clientSig)`

request types

Private and fast smart contracts



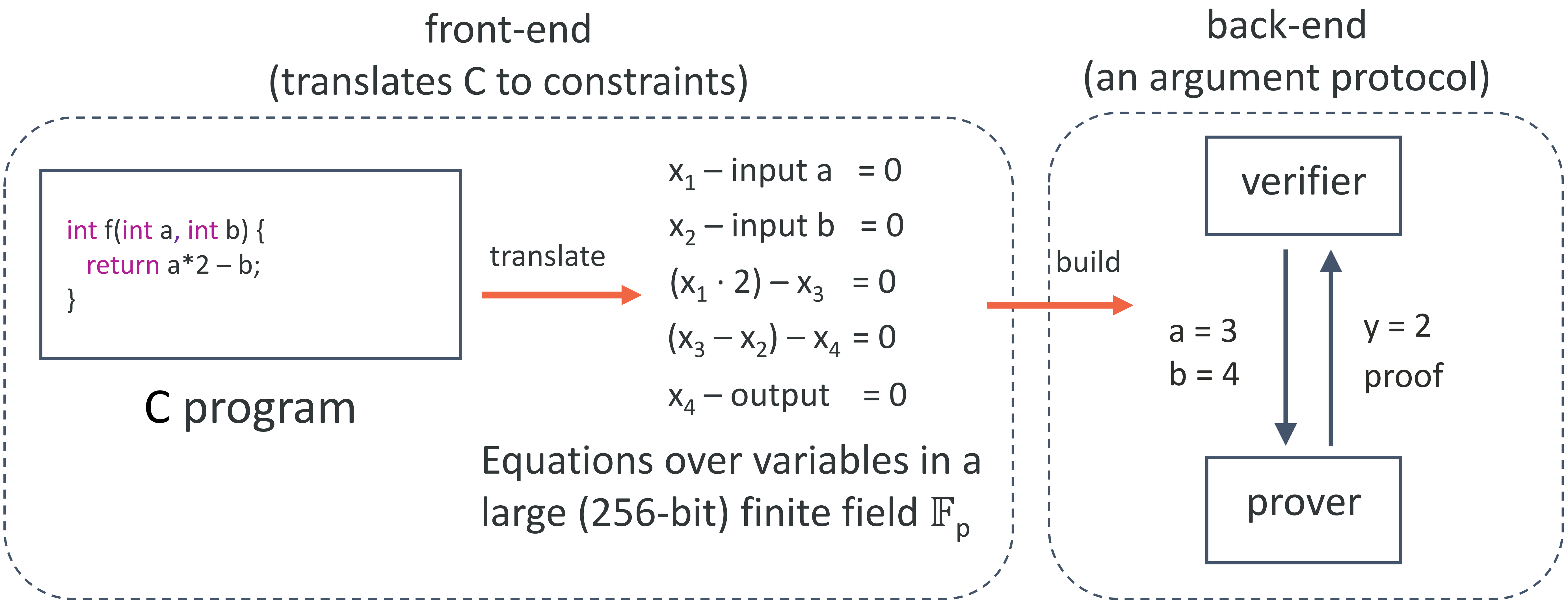
- All transactions and contract state are public → no confidentiality
- Every app-level request must be processed by blockchain→ limits throughput

- Only a succinct trace is public → strong confidentiality
- Ethereum processes a succinct trace → can support high-throughput apps

- Applications
- **Background and overview of Spice**
- Experimental results

A quick overview of Pantry[SOSP13]

- Extends Zaatar[EuroSys13] and Pinocchio[Oakland13] to support state



Zaatar and Pinocchio support a large C subset:

- Arithmetic operations, bitwise operations
- Conditional control flow
- Volatile memory including pointers
- Loops: bound must be known at compile time

Pantry supports state while working in a stateless model---by using cryptographic hashes (a folklore idea)

How does Pantry support state?

Key idea: name data blocks with their short cryptographic hashes

```
    return a*2 - b;
}
```

Cost of a key-value store operation: logarithmic in the size of state

Concretely: several minutes of CPU-time (10^6 KV pairs)

If prover supplies invalid state the assert will fail

Pantry builds a key-value store using this idea: treat hashes as pointers to data and construct a (Merkle) tree

Spice in a nutshell

Core idea: Use a set data structure [Blum et al. FOCS91, Clarke et al. ASIACRYPT03, Arasu et al. SIGMOD17] instead of a tree

- Key-value store op = add an element to a set
- Costs = constant-time (amortized) vs. logarithmic

Spice in a nutshell

Core idea: Use a set data structure [Blum et al. FOCS91, Clarke et al. ASIACRYPT03, Arasu et al. SIGMOD17] instead of a tree

- Key-value store op = add an element to a set
- Costs = constant-time (amortized) vs. logarithmic

However: A naïve instantiation is slower than tree-based approach

- Spice presents an efficient instantiation using ECC (10^6x faster than naïve)
- Spice includes new techniques to support inexpensive transactions
- ….

Implementation of Spice

- 3,000 lines of C atop Pantry [SOSP13] (~15,000 LOC)
- Three apps: 1,500 lines of C

- `init(), insert(Key, Value), put(Key, Value), get(Key), delete(Key)`
- `lock(Key), unlock(Key)`
- `begin_txn(Key[], Value**), end_txn(Key[], Value[])`

- Applications
- Background and overview of Spice
- **Experimental results**

Evaluation questions

1. How does Spice compare with the prior state-of-the-art?
2. What is the end-to-end performance of apps built with Spice?

Evaluation testbed:

- `Azure D64s_v3` instances: 64 vCPUs, 2.4Ghz Intel Xeon, 256 GB RAM, running Ubuntu 17.04

(1) How does Spice compare to prior work?

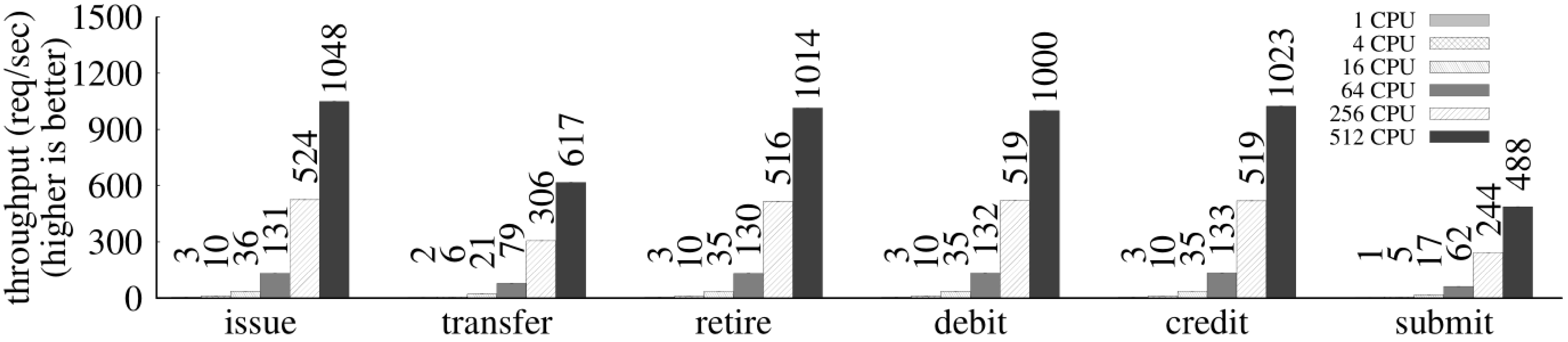
A million key-value pairs

Transactions with a single operation, keys chosen with a uniform distribution

Metric: number of operations/second

	get	put
Pantry [SOSP13]	0.078	0.039
Pantry++	0.15	0.076
Geppetto [Oakland15]	0.002	0.002
Spice (1-thread)	3.3	3.3
Spice (512-threads)	1,250	1,259

(2) End-to-end performance with varying #CPUs



- TPS is 16,000x better than prior state-of-the art (algorithmic + hardware)
- Verification throughput: 15 million proof verifications/second

Summary

- Verifiable state machines is a key tool for the cloud and blockchains

- Spice is a substantial milestone for building verifiable state machines
  - >16,000x better performance (over prior state-of-the-art)
  - Supports real-world applications with thousands of transactions/sec

- We are excited about the many possibilities Spice points to!

Thank you!

Microsoft