BitBill:
Scalable, Robust, Verifiable Peer-to-Peer Billing for Cloud Computing

Li Chen, Kai Chen

SING Lab

Computer Science and Engineering

Hong Kong University of Science and Technology

Trust in the Cloud

Cloud Service Provider

- Did I under-charge the tenant?
- Did I measure the usage correctly?
- Fulfill Service-Level Agreements (SLAs).

Tenant

- Did I over-pay the provider?
- Can I justify my payment and usage?
- "Pay only for what I use"
  - 61% of IT executives and CIOs
- Verify SLAs.

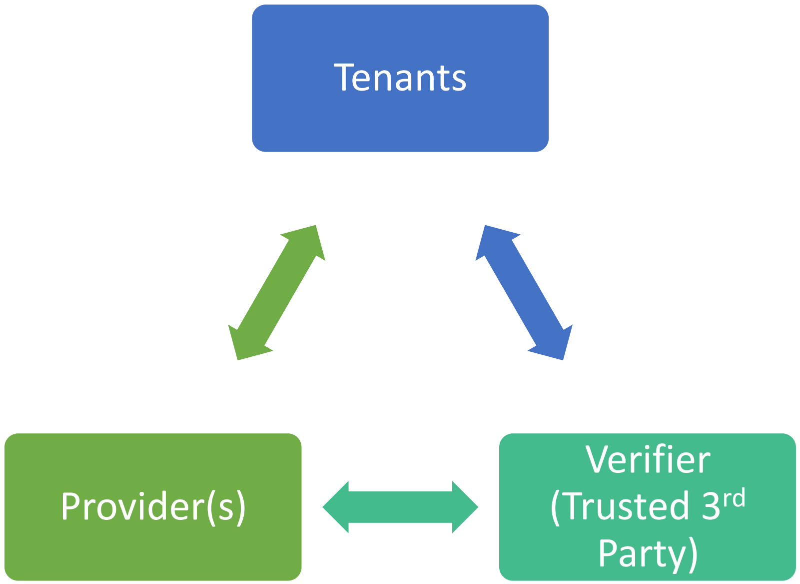
Current Solutions - Industry

- Unconditional trust model
  - All public cloud services in production

- Drawbacks:
  - Tenants have no power in accounting and billing of resource usage after signing the contract.

Current Solutions - Academia

- Third-party trust model:
  - THEMIS
  - ALIBI
  - Verifier
- Drawbacks:
  - Single point of failure
    - Robustness issues
  - Congestive hotspot
    - Scalability issues
  - Assumes verifier's trustworthiness
    - No oversight
    - Another vulnerability



Problem Definition: Billing the Cloud

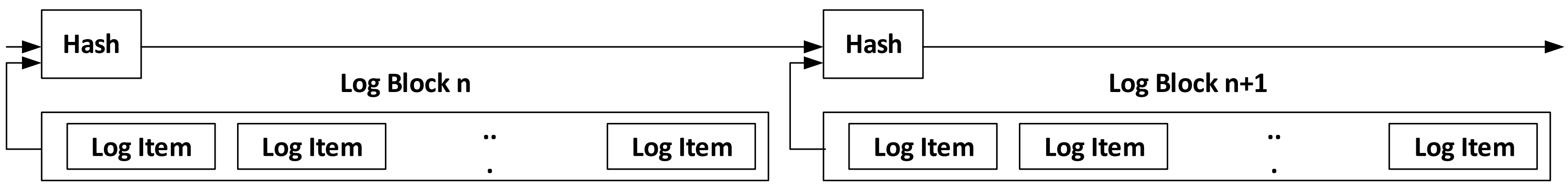
- Mutual verification
- Scalability
- Robustness

- Our Proposal: *A single global history that is maintained and verified by both the tenants and providers*
  - i.e. Keeping a global ledger
  - Public trust model

Public trust model

- No single entity is trusted, but the network is trusted as a whole.
- The tenants and provider(s) form a p2p network that maintains a log of billable events collaboratively.
  - Global history
  - Billing is intuitive based on this mutually maintained log
- Every log of billable event is signed by the corresponding node, and broadcasted to the network.
  - Full history of billable events are known to the BitBill network
- ...But what about malicious peers?
  - Malicious nodes may forge false events to cheat the other nodes.
  - How to keep global state in a untrustworthy environment?
  - The Byzantine Generals Problem
    - Solved by BitCoin-like mechanism

Centralized Solution: Billable event chain



- Centralized "time-stamper"
  - Taking a hash of a block of items to be time-stamped and broadcast the hash.
- The time-stamp proves that the log item must have existed at the time, in order to get into the hash.
- Each timestamp includes the previous timestamp in its hash, forming a chain.
- With each additional timestamp reinforcing the ones before it.

Byzantine Generals Problem

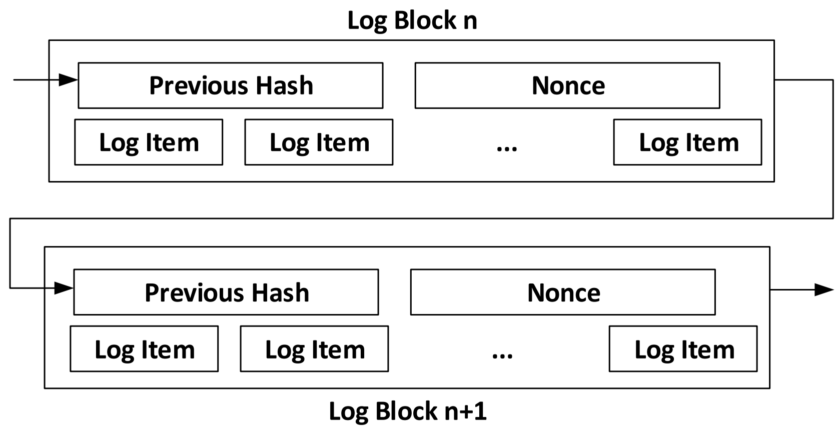
a.k.a, Byzantine fault tolerance, Two-generals problem.

- Reaching agreements with possible existence of malicious peers.
- Peers send/receive messages to/from each other trying to reach an agreement.

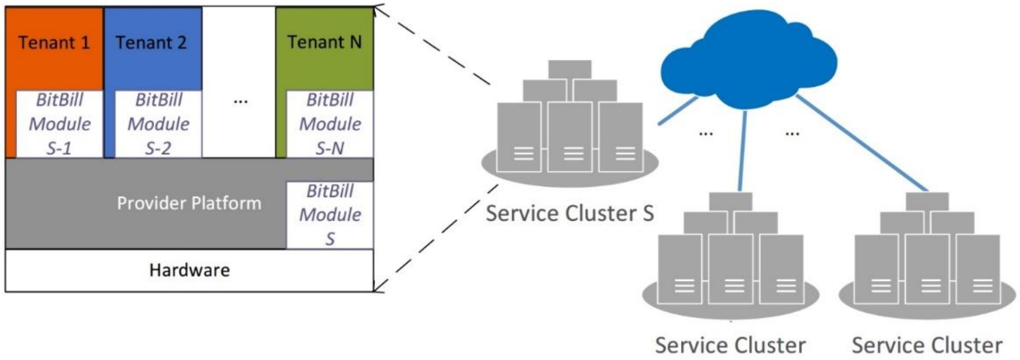
- BitCoin solves this by adding a cost to the announcement
  - So that malicious node cannot send frequently.
  - Only one node is allowed to send, acting like the “time-stamper”.

Decentralize the Time-stamper: Proof-of-Work (PoW) technique

- Proof-of-work is the solution to the problem.
- Adding a cost to the announcement
  - Create insurmountable difficulty for the malicious nodes to forge logs.
- Nodes have to compute a number before sending out a broadcast
  - Calculation takes a long time, so that only one node in the network is able to find it at any given time.



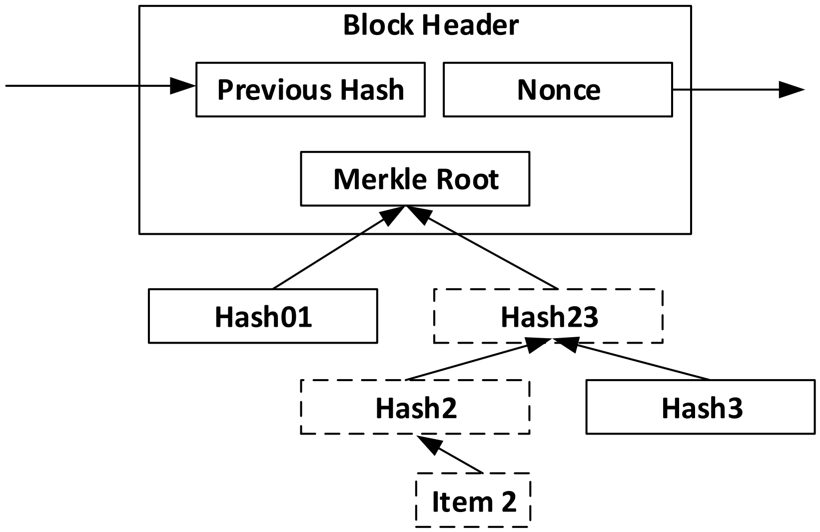
BitBill network operations



1. New usage logs are broadcast to all nodes.
2. Each node collects new usage logs into a block.
3. Each node works on finding a difficult proof-of-work for its block.
4. When a node finds a proof-of-work, it broadcasts the block to all nodes.
5. Nodes accept the block only if all logs in it are valid.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Existence of log items

- Merkle Tree
  - every non-leaf node is labelled with the hash of the labels of its children.
- Verifying
  - Find block headers of the longest chain
  - Obtain the Merkle branch linking the transaction to the block it's time-stamped in.
- Existence of log item:
  - A network node has accepted it
  - blocks added after it further confirm the network has accepted it.



BitBill verification

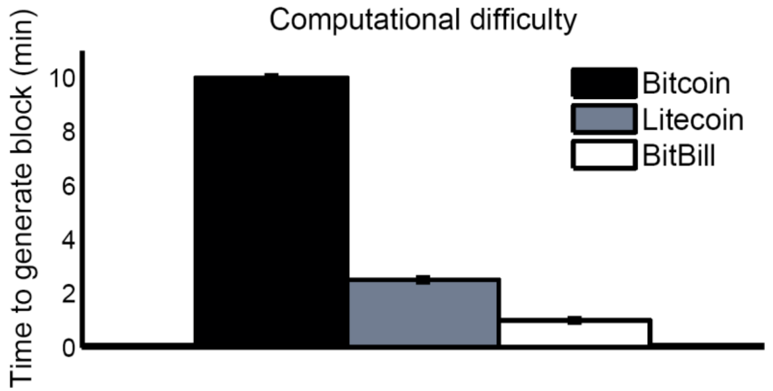
- BitBill does one thing and one thing only:
  - Keeping a global ledger (history of events)
- Identify and resolve these conflicts at the nodes.

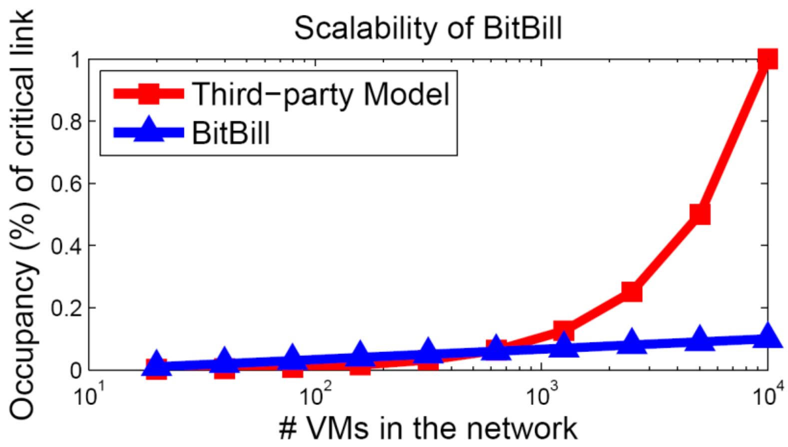
- Verification is to resolve conflicts in the recorded events
  - A CPU cycle cannot be used concurrently by 2 tenants.
  - Instantaneous bandwidth cannot exceed link capacity.
  - Aggregated memory usage cannot exceed physical limit.

Resource to enable BitBill

Influencing factors:

- On end-host:
  - Frequency of broadcast
  - Difficulty of PoW
  - Granularity of events
- On network
  - Number of nodes
  - Frequency of broadcast





Discussions

- **Privacy**
  - decoupling public key and identity
- **Deployment**
  - preinstalled software package
- **Resource monitoring**
  - granularity

Summary

- We introduce a new trust model, the public trust, with regard to mutually verifiable billing in the cloud.
- Novel use of the Bitcoin-like mechanism to deal with the Byzantine Generals Problem that comes with this model.
- We design a scalable and robust billing and accounting solution, BitBill.

Thank you!

- Q&A