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

Li Chen
CSE, HKUST
lchenad@ust.hk

Kai Chen
CSE, HKUST
kaichen@ust.hk

Abstract

Accounting and billing of cloud resources is vital for the operation of cloud service providers and their tenants. In this paper, we categorize the trust models of current industrial and academic cloud billing solutions, and discuss the problems with these models in terms of degree of trust, scalability and robustness. Based on the analysis, we propose a novel public trust model to ensure natural and intuitive verification of billable events in the cloud. Leveraging a Bitcoin-like mechanism, we design BitBill, a scalable, robust and mutually verifiable billing system for cloud computing. Our initial results show that BitBill has significantly better scalability (supporting 10x concurrent tenants using the billing service) than the state-of-the-art third-party centralized billing system.

1 Introduction

The ease of dynamic deployment and scaling of computation service motivates the adoption of cloud computing. Tenants of cloud computing services do not have to maintain dedicated infrastructure, and are promised the ability to flexibly and dynamically adjust the computational tasks as their business grow with little administrative and capital overhead [1, 2]. However, billing the cloud is still a major concern for both the tenants and providers of the cloud.

Trust issues between the tenants and service providers are the main obstacles hindering the wide adoption of cloud computing. Tenants benefit from knowing to what extent a provider delivers the promised service. For example, 61% of IT executives and CIOs rated the "pay only for what you use" as a very important perceived benefit of the cloud model [2]. However, as the tenants have little or no exposure to the actual resource consumption of the cloud, they are unable to justify their charges with their computational tasks. For providers, it is also vital to let the tenants know that they are not overcharged, so that they will continue to use the provider's service. Providers also suffer from being unable to accurately account for the usage of every type of resources by the tenants. For instance, memory bandwidth and I/O stress cannot be precisely justified [3, 9, 15], which leads to uncertainty and inaccuracy, and providers may lose revenue due to undercharging. For the benefit of both the tenants and providers, a mutually verifiable log of resource usage and service agreements should be kept by both the provider and tenants, so that the billing is accurate for provider, and convincing for tenants.

However, resource usage accounting should not only limited to one tenant and its provider. All tenants of a provider are involved when the provider provisions a piece of resource to one tenant. Public cloud resources are shared, and providers reduce their capital and management costs by statistically multiplexing tenants on the same infrastructure using virtualization technology [13]. For example, Amazon's AWS[1] uses an over-subscription model to share a same piece of hardware between multiple tenants [6]. Provider's double billing of resource means that the performance isolation [10] between the tenants cannot be guaranteed. While it potentially violates the contract between tenant and provider, double billing can hardly be detected by any single affected tenant. In existing literature on billing, there lacks a trustworthy mechanism that guards tenants against such breach of contract.

To address the above trust issues, we start with summarizing the existing two trust models between the tenants and provider(s), and then discuss their problems. The first model is unconditional trust, which is adopted by all cloud providers (e.g. Amazon AWS, Microsoft Azure[2], Rackspace[3]) in production. With this model, the tenants have no power in the accounting and billing of

[1] http://aws.amazon.com/ec2
[2] http://www.windowsazure.com
[3] http://www.rackspace.com

the resource usage, and they trust the provider to accurately calculate the resource usage, and to bill according to the contracts. Since there is no way for tenants to examine the actual usage, the tenants' IT executives have difficulty in justifying the costs.

The second model is third-party trust, which introduces an outside authority to ensure that the accounting and billing are mutually verifiable. This model has been implemented for Platform-as-a-Service (PaaS) paradigm [13] and Infrastructure-as-a-Service (IaaS) paradigm [2, 12]. The designated third party verifies the resource usage and compliance with the agreement, and billing incurred can be verified by both the tenant and the provider. The problem with this trust model is scalability and robustness. These solutions (i.e. [2, 12, 13]) all propose to use a centralized third party for verification, thus introducing a single-point-of-failure (SPOF) and a congestive hotspot. The scalability of the cloud suffers as the processing power of the third party limits the capacity of the cloud. The robustness of the cloud is also endangered: if the central verifier or the link to the verifier fails, the verification service will no longer be available, and the tenants again have only the provider(s) to trust unconditionally.

Therefore, to achieve a mutually verifiable, robust and scalable billing solution for the cloud, a new trust model is needed. Our position in this paper is that, by forming a cryptographic peer-to-peer (p2p) network of the tenants and provider(s), we can adopt the Bitcoin-like [11] mechanism to design a billing system, BitBill, for the cloud, so that the aforementioned desirable properties can be achieved.

BitBill establishes a *public trust model* where the nodes in the network do not trust any single entity, but the network as a whole. Inspired by Bitcoin, BitBill provides an elegant solution to the problem of keeping a global state in an untrustworthy environment, i.e. thew well-known Byzantine Generals Problem [5]. In the context of cloud accounting and billing, this global state is a log of all the resource provision and usage of all the tenants and the provider(s), and this state (or log) serves as the verified record for billing. With collaboration among tenants in one resource pool (e.g. tenants using a same physical rack), the tenants and provider(s) verify and fully agree on one log of events, based on which billing is straightforward. This solution is scalable and robust, since the message is broadcasted on a best effort basis and does not assume any network condition. In addition, double billing can easily be detected, because any piece of the resource pool allocated to some tenants affects the available resource of the other tenants in the same resource pool.

It is important to note the BitBill is orthogonal to pricing or state monitoring in the cloud. Our scope is limited to keeping a global "ledger" of the resource-related activities in the cloud. We build on previous state monitoring solutions [2, 7, 13] to design BitBill.

BitBill has the following advantages over existing billing systems and architectures:

- BitBill introduces a new trust model, the public trust model, with regard to mutually verifiable billing in the cloud.
- We make 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 for the cloud with BitBill by implementing the trust model in a p2p manner. Our initial results show that BitBill is significantly more scalable (supporting 10x concurrent tenants) than the state-of-the-art third-party centralized billing system.

We discuss related work in Section 2, and introduce the BitBill system in Section 3. We perform numerical simulations of BitBill in Section 4 to examine its operational overhead. Section 5 discusses further issue of BitBill, and we conclude the paper in Section 6 with a summary and a roadmap of our future work.

2 Related Work

Current cloud services in industry all used unconditional trust model for billing, while current works in academia focus on the third-party trust model. The following three proposed solutions are most related to our BitBill design.

- ALIBI [2] defines three types of integrity: image, execution and accounting. ALIBI uses nested virtualization to place a trusted "Observer" at the highest privilege level underneath the providers platform software and all customer instances.
- A high level systematic solution for verifiable resource accounting is proposed [13]. A "verifier" service is defined on an abstract level, and practical approximations and relaxations are designed for realization of this conceptual design.
- THEMIS [12] explores how billing of transactions in cloud computing can be mutually verified by both tenant and provider with small computational overhead. THEMIS uses a cloud notary authority to oversee the resource consumption, which makes future resolutions of dispute more acceptable and objective.

The common feature in all three proposals is that there is a third-party that supervises the resource allocation and/or billing to achieve the desired properties of billing for the cloud. The problems with this feature are threefold.

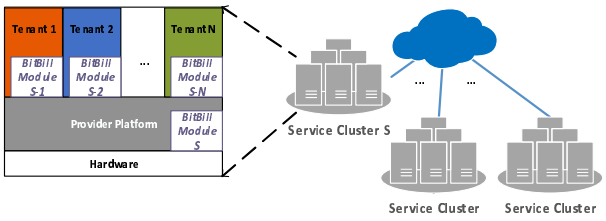

Figure 1: BitBill system concept

First, the trust between the third-party and tenants (or provider) must be established. THEMIS provides the required degree of trust and security by simulating the Public Key Infrastructure [4]. However, such mechanism only verifies the transaction, but does not protect the tenants against the over-provisioning of resources by provider. The tenants have no choice but to trust the ruling of the third-party in terms of billing, which is fundamentally the same as unconditional trust model.

Second, it makes the billing system inherently vulnerable to potential attacks. As all transactions have to be processed by the third party, if the third-party is down, the billing cannot continue. Keeping the billing always available is an integral part of provider's operation, therefore a single-point-of-failure of the trusted third-party verifier may bring down the providers using its service.

Lastly, scalability of the verification of the transactions becomes an important issue when the number of tenants grows. The trusted third-party is a congestive hotspot as all billing requests have to be processed by it. Every virtual machine of every tenant needs to have its resource usage verified, which poses heavy load on the verifier as we illustrated in the evaluation section later.

We propose BitBill as a cloud billing system that is based on a novel public trust model, so as to avoid the aforementioned drawbacks.

3 BitBill Overview

3.1 Public trust model

The problem we intend to deal with comes with the public trust model. In this model, no single entity is trustworthy, but the network is trusted as a whole, given the only condition that the malicious participants control less computational power than the honest ones. In BitBill network for the IaaS paradigm, as shown in Figure 1, the tenants and provider(s) form a p2p network that maintains a log of billable events collaboratively using a Bitcoin-like protocol described below. With BitBill, we can achieve mutually-verifiable, robust and scalable billing of cloud resources.

Before going into details of the design of BitBill, we first clarify the terms and concepts involved. Cloud network resources include, be not limited to, the CPU cycles, memory bandwidth, memory size, I/O bandwidth, network bandwidth. The definition of billable events depends on requirements of different providers, and the events usually include usage of CPU cycles, network bandwidths, storage I/O, etc. Generally speaking, a billable event should be the consumption of cloud resources.

3.2 Proof-of-work technique

The over-subscription problem is that the tenant cannot be sure that the service provider did not "double bill" the resource, and therefore its performance may be influence by the performance of other tenants. Existing proposals have introduced third-party to solve this problem, but this centralized solution also introduces problem of trust issues, scalability and robustness. In BitBill, we need all the participants to agree on a single history of the order of events. Thus every billable event has to be announced. For every resource consumption event by the tenant and resource allocation event by the provider, a log message signed by the corresponding party is broadcasted to the network. In this way, full history of billable events are known to the BitBill network, and therefore the first provisioning of the resource can be found. However, malicious nodes may forge false events to cheat the other nodes, and to counter this, we employ the Proof-of-work (PoW) technique used by Bitcoin [11].

The problem caused by the malicious node is formally the Byzantine Generals Problem [5], and Bitcoin solves it by adding a cost to the announcement, so as to create insurmountable difficulty for the malicious nodes to forge transcations. The cost is designed such that the rate of confirmed announcement is slowed down, and the inherent randomness of the cost ensures that only one participants will be able to broadcast at a time. The randomness is due to the calculation of a random hash function. In BitBill, as shown in Figure 2, the input of the hash function is the entire hashed history of the billable events up to the current point in time and a random number ("Nonce"). Only a hash value where the first 5 characters are zero is accepted by BitBill as the "proof of work". Since a hash output is easy to verify but hard to find the corresponding input, the other nodes can examine the announcement quickly, and continue to work on the subsequent announcement. Later blocks are chained after it, validating the history of events.

Note that BitBill uses a much easier PoW than Bitcoin, which requires the hash to have the initial 13 characters to be 0 [11]. We illustrate the computational difficulty in the evaluation section. BitBill use a easier version because the logging events are frequently occurring in a cloud, therefore we need higher announcement frequency than bitcoin.

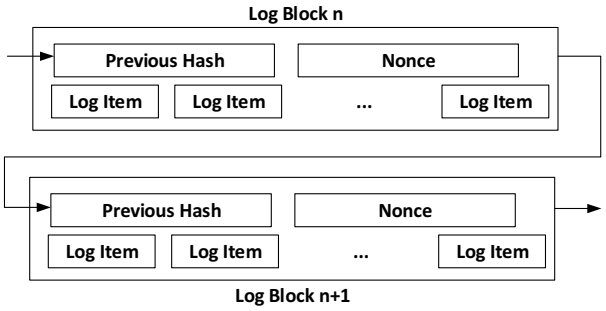

Figure 2: Proof-of-work Mechanism

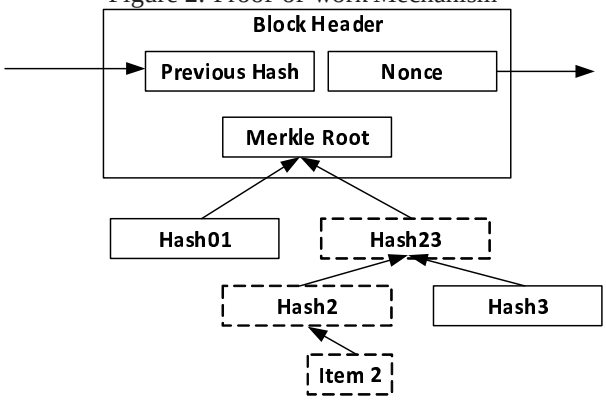

Figure 3: Merkle tree and billing verification

3.3 Verification of events

BitBill uses a Merkle tree [8] to store and verify events efficiently. Merkle tree is a tree in which every non-leaf node is labelled with the hash of the labels of its children nodes. Merkle tree is suitable for BitBill because they allow efficient and secure verification of the contents of larger data structures.

In Figure 3, billable Item 2 is being verified by a node through the Merkle root in the header of the current longest block-chain known to the node.

3.4 BitBill node operation

The BitBill network runs as follows:

1. Broadcast new transactions are to the network.
2. Each node collects new transactions they receive 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 transactions 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.

BitBill nodes will consider the longest chain to be the correct one, and they will keep working on extending only the longest one. For the case that two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next PoW is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

New billable event broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, a block will eventually collect them. Block broadcasts are also fault tolerant in case of dropped messages. If a node does not receive a block announcement, it will realizes it missed the block when it receives the subsequent block(s). Upon realization, the node will request the missed block.

With a single history of billable events known, the billing can be done easily. For each settlement between the provider and the tenant, the billing is done according to the longest block chain maintained in the BitBill network. The tenants and the provider will not have any dispute, since there is only one history of record.

4 Evaluation

As every tenant have to run BitBill to participate in the billing system, it is important that BitBill does not incur unnecessary overhead during its operation. To examine this, we perform numerical simulations to evaluate the computational overhead and network overhead of BitBill. In the evaluation section, we focus on the computational load of BitBill and the scalability (determined by the network overhead).

For Bitcoin-like protocols, the computational difficulty is usually measured by the average time required to generate a block. This can be adjusted by either changing the hash function (e.g. Litecoin [14]), or adjusting the parameters to relax the requirement of accepting a hash. We chose the latter approach for BitBill, and the difficulty difference is shown in Figure 4. Bitcoin takes 10 minutes to generate one block, Litecoin 2.5 minutes and BitBill takes less than 1 minute. Smaller block generation time is suitable for a small p2p network in the cloud compared to the Bitcoin network, which has hundreds of thousand of nodes.

We continue to analyze the scalability of BitBill compared with a centralized third-party verifier. For BitBill, the critical link is the one with highest volume of broadcasted messages, while for the third-party model, the critical link is the one connected to the verifier. In the simulation, we vary the number of VMs in the network from 10 to 10^4, and we assumer the VMs send their messages (usage reports) every 5 seconds. Each message is 500Kb in size. We set all the link bandwidth to be 1Gbps.

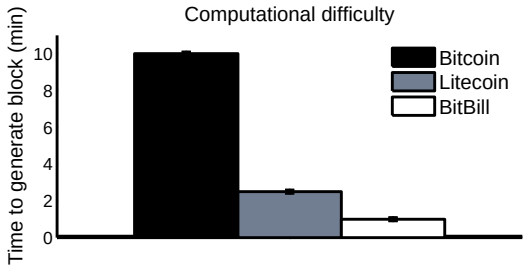

Figure 4: Comparison of difficulty of different Bitcoin-based protocols

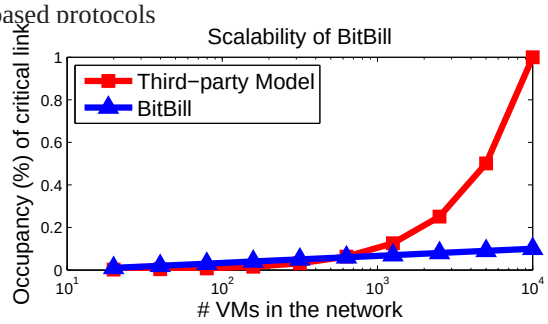

Figure 5: BitBill scalability compared against a centralized third-party verifier

In Figure 5, we plot the occupancy of bandwidth of the critical link in both models measured in our numerical simulation. When the number of VM is 10^4, the link occupancy is almost 100% for the centralized model, and therefore the billing service cannot support anymore VMs. Whereas the occupancy of BitBill is kept under 10% for the same amount of VMs, and the maximum number of VMs supported is 10 times that of state-of-the-art centralized mechanism. We can conclude that BitBill is a billing system that is much more scalable, i.e. it can accommodate much more VMs than any centralized mechanism.

5 Discussion

In this section we discuss several issues regarding the implication of public trust model, deployment and implementation issues about BitBill.

Management layer mechanism to protect performance isolation Due to BitBill's model of public trust, it is also a natural management level mechanism to protect against performance interference [10] between tenants. Statistical multiplexing is the source of such interference, and BitBill can detect the "double billing" with the collaboration among the tenants. This management layer performance isolation can assist the isolation mechanism on other layers of the cloud.

Privacy BitBill keeps tenant privacy by making public keys anonymous. The public can see only the amount of resource assigned to some node, but cannot link this piece of information to anyone. Therefore the privacy of the tenants is protected. Any tenant in the network can only know the resource usage of a node, but the node's identity can not be known.

Deployment BitBill can be preinstalled by provider as a software package along with the VM. A trusted hash of the package can be used by the tenants to verify the integrity of the package. Tenant can also install their own BitBill module on their VMs, and participate in the network voluntarily. In this way, the trust model is flexible, since unconditional, third-party and public trust models can co-exist in the same cloud.

Resource monitoring BitBill can build upon and extend monitoring tools in both industry and academia [7]. However, these solutions are provider-centric, and focus on resource information collection efficiency. The tenants does not benefit from them. BitBill, by establishing a public trust model, empowers the tenants so that they can gather evidence of correctness from the broadcasts of the other tenants. ALIBI is a first step in incorporating resource monitoring into billing, and we can built on it to implement a better trust model.

Security issues BitBill defends against malicious insider nodes with PoW mechanism. The attacker will have to redo all the PoW in order to forge a history of events that can be accepted by the other nodes. Like Bitcoin [11], BitBill is essentially one-CPU-one-vote, and the system is secure as long as the majority of the nodes are honest.

6 Conclusion and future work

In this paper we propose a novel trust model, public trust, for billing in the cloud. With a distributed mechanism, BitBill maintains a global log of billable events in a completely untrustworthy environment. Due to its distributed nature, BitBill is both scalable and robust to network failure, which is unaddressed in existing literature.

For future work, we intend to implement BitBill as a kernel module, and deploy BitBill on a real testbed. We understand that the solution that we proposed are preliminary, and some practical issues must be resolved. The issues include: measuring and reducing the network I/O overhead, reducing the computational load of the module, minimizing the resource monitoring overhead of the module, etc. We intend to address these issues during the implementation and experiments in the future.

Acknowledgement

This work is supported by China 973 Program Grant 2014CB340303, and HKUST Grant REC12EG07. We thank the HotCloud anonymous reviewers for their comments.

References

[1] BOUCHENAK, S., CHOCKLER, G., CHOCKLER, H., GHEORGHE, G., SANTOS, N., AND SHRAER, A. Verifying cloud services: Present and future. *SIGOPS Oper. Syst. Rev. 47*, 2 (July 2013), 6–19.

[2] CHEN, C., MANIATIS, P., PERRIG, A., VASUDEVAN, A., AND SEKAR, V. Towards verifiable resource accounting for outsourced computation. *SIGPLAN Not. 48*, 7 (Mar. 2013), 167–178.

[3] IYER, R., ILLIKKAL, R., ZHAO, L., NEWELL, D., AND MOSES, J. Virtual platform architectures for resource metering in datacenters. *SIGMETRICS Perform. Eval. Rev. 37*, 2 (Oct. 2009), 89–90.

[4] KUHN, D. R., HU, V., POLK, W. T., AND CHANG, S.-J. H. Sp 800-32. introduction to public key technology and the federal pki infrastructure. Tech. rep., Gaithersburg, MD, United States, 2001.

[5] LAMPORT, L., SHOSTAK, R., AND PEASE, M. The byzantine generals problem. *ACM Trans. Program. Lang. Syst. 4*, 3 (July 1982), 382–401.

[6] LE-QUOC, A., FIEDLER, M., AND CABANILLA, C. The top 5 aws ec2 performance problems.

[7] MENG, S., IYENGAR, A. K., ROUVELLOU, I. M., LIU, L., LEE, K., PALANISAMY, B., AND TANG, Y. Reliable state monitoring in cloud datacenters. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing* (Washington, DC, USA, 2012), CLOUD '12, IEEE Computer Society, pp. 951–958.

[8] MERKLE, R. C. A digital signature based on a conventional encryption function. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology* (London, UK, UK, 1988), CRYPTO '87, Springer-Verlag, pp. 369–378.

[9] MOGUL, J. C. Operating systems should support business change. In *Proceedings of the 10th Conference on Hot Topics in Operating Systems - Volume 10* (Berkeley, CA, USA, 2005), HOTOS'05, USENIX Association, pp. 8–8.

[10] MOGUL, J. C., AND POPA, L. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Computer Communication Review 42*, 5 (2012), 44–48.

[11] NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. *Consulted 1* (2008), 2012.

[12] PARK, K.-W., HAN, J., CHUNG, J., AND PARK, K. H. Themis: A mutually verifiable billing system for the cloud computing environment. *Services Computing, IEEE Transactions on 6*, 3 (2013), 300–313.

[13] SEKAR, V., AND MANIATIS, P. Verifiable resource accounting for cloud computing services. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop* (New York, NY, USA, 2011), CCSW '11, ACM, pp. 21–26.

[14] STEVENSON, J. *Getting started with Litecoins (after Bitcoin)*. John Stevenson, 2013.

[15] WACHS, M., XU, L., KANEVSKY, A., AND GANGER, G. R. Exertion-based billing for cloud storage access. In *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing* (Berkeley, CA, USA, 2011), HotCloud'11, USENIX Association, pp. 7–7.