The Byzantine Empire in the Intercloud

Marko Vukolić

IBM Research - Zurich
CH-8803 Rüschlikon, Switzerland
mvu@zurich.ibm.com

Abstract

The relevance of Byzantine fault tolerance in the context of cloud computing has been questioned [3]. While arguments against Byzantine fault tolerance seemingly makes sense in the context of a single cloud, i.e., a large-scale cloud infrastructure that resides under control of a single, typically commercial provider, these arguments are less obvious in a broader context of the *Intercloud*, i.e., a cloud of multiple, independent clouds.

In this paper, we start from commonly acknowledged issues that impede the adoption of Byzantine fault tolerance within a single cloud, and argue that many of these issues fade when Byzantine fault tolerance in the Intercloud is considered.

1 Introduction

In dependable computing, the term "Byzantine" fault is a catch-all for any type of software or hardware fault. It is frequently associated with malicious behavior and intrusion tolerance; however, it also encompasses spurious and arbitrary faults as well as conventional crash faults. Hence, it is not surprising that Byzantine fault tolerance (BFT) has received a lot of research attention ever since the concept of a Byzantine fault was introduced three decades ago [24, 26], as it promises dependable systems that tolerate any type of misbehavior.

Despite this attention and appealing promises, BFT suffers from very limited practical adoption [21]. Notable exceptions include a modest number of safety critical embedded and real-time systems, e.g., in the aerospace industry [14]. However, in the more conventional distributed systems, BFT remains sidelined.

In particular, the relevance of BFT has been questioned in the context of cloud computing, which is arguably one of the main drivers of the distributed systems agenda in the last few years. Practitioners, whose daily bread-and-butter is dealing with (very) large-scale distributed systems that propel modern cloud services, describe BFT as of "purely academic interest" for a cloud [3]. Such a lack of interest for BFT within a cloud is in sharp contrast to the popularity of mechanisms for tolerating crash faults which are routinely employed in such large-scale systems. Furthermore, the wealth of BFT research that is doing hard work to make BFT practical and to bring the performance of BFT protocols close to that of their crash-tolerant counterparts (e.g., [8, 12, 19]), appears incapable of spurring the interest for BFT applications.

Factors that impede wider adoption of BFT arguably include notorious difficulties in design, implementation, proofs and even simple understanding of BFT protocols, especially when BFT protocols also have to face asynchrony [17]. However, while relevant, this seems not to be the core of the problem. In fact, it seems that there are strong reasons to believe that even if the BFT community came up with provably optimal and best-possible protocols, such BFT protocols would still fail to make it into the large scale cloud; we discuss these reasons and arguments in more details in Section 2.

However, one should beware of not seeing the forest for the trees: the cloud computing landscape obviously does not end with a single cloud. The sky is very cloudy these days, with many clouds that come in different shapes and colors, forming the cloud of clouds, or simply, the *Intercloud*[1]. These many shapes and colors reflect different implementations and administrative domains, which are like a dream come true for a designer of dependable and fault-tolerant systems, e.g., in terms of failure-independence. There are already some very early protocols that are designed for and deployed in the Intercloud [1, 4, 6] that leverage multiple clouds to add value to the client beyond the dependability guarantees of any individual cloud. The Intercloud can be simply seen as the second layer in the cloud computing stack and it calls for considerably different protocols than the "bottom" layer, i.e., the *inner-cloud* layer [6].

In this paper, we develop the argument around the observation made in [7], that it is in this second layer, in the Intercloud, where the modern Byzantine Empire might find its place. More specifically, we argue (Section 3) why the Intercloud is suitable for BFT, but also (and perhaps more importantly) why BFT is suitable for the Intercloud.

2 Limitations of BFT for the Inner-cloud

Today's inner-cloud is characterized by large-scale distributed systems, such as Amazon's Dynamo [13], Google's File System (GFS) [16] and BigTable [10], Yahoo's Zookeeper [20] or Facebook's Cassandra [2], running in a single administrative domain and powering some of the most popular services in existence. While these protocols deal with crash-failures routinely, there is no (known) trace of BFT inside them. Let us hypothetically consider the best possible and by all means optimal BFT protocol for a given task. The following arguments would still impede the adoption of such a protocol in the inner-cloud.

Inherent cost. The inherent cost of BFT in terms of number of replicas needed to tolerate failures is often cited as one of its major drawbacks. Consider state-machine replication (SMR) [5, 30] as a classical example: crash and asynchrony-tolerant implementations like Paxos [22], require at least $2f+1$ nodes to tolerate f failures, whereas in the case of BFT Paxos $3f+1$ nodes are required [23]. While assuming small trusted hardware components inside nodes can eliminate this overhead (see e.g., [25, 32] for recent contributions), trusted architectures have yet to be embraced more widely.

Failure independence. All fault-tolerant protocols, including BFT, require some level of failure independence [31]. However, in the case of BFT, this level is particularly high. Consider for example a Byzantine failure that results from a malicious exploit of a vulnerability of a given node. To prevent such a failure to propagate to other nodes, one must assume different implementations, different operating systems, separate administrative domains and even different hardware. This should be complemented by the diversity required for independent crash-failures, which includes

[1]`http://www.google.com/search?q=intercloud`

[2]The Apache Cassandra Project. `http://cassandra.apache.org/`

different power supplies and possibly even geographical diversity. "In-house" maintenance of the level of infrastructural diversity that BFT requires is prohibitively expensive even for the large-scale cloud providers [21].

Does the model match the threats? Clearly, the above issues which are in the end associated with the cost of deploying BFT, would not be as critical if the Byzantine model would be adequate for the threats within a cloud. However, the Byzantine failure model is often seen as too general, allowing for any type of failures, notably malicious ones. Although attacks on well-known cloud providers occur on a regular basis and sometimes have success [3], exploits that would allow the attacker to hijack the critical part of the internal cloud infrastructure are yet to be seen. Such infrastructure chooses intrusion prevention and detection over intrusion tolerance, and is carefully protected and isolated from the "outside" world. Consider, for example, Google's Chubby [9], a distributed locking and low-volume storage service based on Paxos (and a rare example of SMR within a cloud). Chubby is a shielded service offered to very selected clients such as Google's own GFS and BigTable nodes which do not present a realistic misbehavior threat for Chubby. The same is true for other critical parts of the inner-cloud (e.g., Yahoo Zookeeper) — these critical services that could potentially profit from BFT simply seem not to need it since they typically run in protected environments.

Rare non-crash failures in the inner-cloud call for a failure model more restricted than Byzantine, that would ideally come with a lower cost. However, devising such an adequate failure model for such a single administrative domain, that would be more restricted than Byzantine yet allow for some randomized behavior and a few bugs, is not obvious. Other popular models in the BFT community, like BAR (in which nodes can be Byzantine, altruistic or rational) [2] were devised with a different setting in mind (e.g., cooperative services in peer-to-peer networks) and are clearly not applicable here. This model/threat mismatch seems to be one of the main factors that keep BFT out of the inner-cloud picture.

3 BFT in the Intercloud

Despite their high availability goals, individual cloud services are still subject to outages (see e.g., [7]). Moreover, studies show that over 80% of company executives "fear security threats and loss of control of data and systems".[4] One way to approach these issues is to leverage the Intercloud, i.e., the cloud of clouds. The key-high level idea here is essentially to distribute the security, trust and reliability across different cloud providers to improve on the offerings of any individual one. Early examples of such dependable systems in the Intercloud target reliable distributed storage [11] some of which employ a subset of BFT techniques. For example, RACS (Redundant Array of Cloud Storage) [1] casts RAID into the Intercloud setting to improve availability of data and protect against vendor lock-in. HAIL (High Availability and Integrity Layer) [4] leverages multiple clouds to boost the integrity and availability of the data stored in the clouds. ICStore (Intercloud storage) [6] goes a step beyond RACS and HAIL in terms of dependability, and aims to allow *a la carte* modular combinations of individual layers dedicated to boosting confidentiality, integrity,

[3]Official Google Blog: A new approach to China. `http://googleblog.blogspot.com/2010/01/new-approach-to-china.html`

[4]Survey: Cloud computing 'no hype', but fear of security and control slowing adoption. `http://www.circleid.com/posts/20090226_cloud_computing_hype_security/`

reliability and consistency of the data stored in the clouds. In the following, we argue why the Intercloud is a big opportunity for BFT in general and what BFT applications could appear early-on in this novel context.

Unprecedented failure independence. The Intercloud offers different implementations of semantically similar services along with the unprecedented failure independence. Intuitively, there is much less failure dependence between a virtual machine hosted by Amazon EC2 and another one by Rackspace Cloud Servers than between two machines in a given local cluster [18]. These two virtual machines come in different administrative domains with different proprietary cloud architecture and middleware, possibly different operating systems, not to mention different geographical locations and power supplies. And, best of all, this diversity comes to a BFT designer essentially for free — the maintenance of this diversity remains in the hands of individual cloud providers. Practically, BFT designers and developers are left worrying only about careful programming of their own protocols. The need for n-version programming might persist, but this becomes easier with the use of right abstractions and the modular approach to the problem [17, 28, 31]. Overall, the inherent diversity of the Intercloud promises to significantly reduce the costs of deploying and maintaining BFT protocols.

The threat calls for BFT. Furthermore, the threat that a virtual machine accessible from anywhere faces is arguably different from the one faced by a Paxos node running deeply within the protected environment of a provider's inner-cloud infrastructure. For example, the increased concern over a virtual machine security stems from the possibility of the compromise of the credentials needed to access the virtual machine remotely, but also from the issues related to cloud multi-tenancy (see e.g., [27]). Hence, the Byzantine model seems to reasonably fit the threat faced by a virtual machine hosted by the cloud; users, including company executives, might sleep better knowing that their (Inter)cloud service is dependable and protected from attack on any individual cloud.

How about inherent cost? On the other hand, the Intercloud clearly does not reduce the inherent cost overhead of BFT compared to crash fault-tolerance in terms of a number of replicas. However, while putting trusted hardware components into virtual machines might not be as simple as putting them into physical machines, there is increasing effort to address those aspects of cloud computing (e.g., Terra [15] and TCCB [29]). The Intercloud and BFT will inevitably profit from these efforts that aim to strengthen the security guarantees of individual clouds; these efforts would allow established BFT protocols that leverage trusted architectures to shine in a new light of the Intercloud. An unresolved issue would still be for industry to embrace such architectures. However, given the level of customers' concerns over cloud security, we might witness market differentiation bootstrapping the process of the adoption of trusted cloud architectures.

What early-bird applications? The Intercloud indeed seems to be a promising candidate for an answer to the question on "where" (in the cloud context) should BFT be deployed [21]. On the other hand, to answer questions "why" and "when" [21], we still need to identify an application that will illustrate the full benefits of BFT in the Intercloud and promote its adoption. This might be BFT state machine replication in an implementation of a very reliable low volume storage service, along the lines of Chubby. This service could be shared among different clients coming from, e.g., small private clouds, and for which the clients would be prepared to pay the premium in order not to deal with complexities and the cost of the maintenance of such a service “in-house”. Such a low volume storage application in the Intercloud could in fact need BFT since, unlike Chubby, neither its clients nor its server nodes scattered among the clouds can be fully trusted. Furthermore, BFT could be used in the Intercloud to mask possible inconsistencies inherent to the eventually consistent semantics of a highly available cloud [33]. In this approach an inconsistent reply from an otherwise correct cloud service could be masked as a Byzantine and the Intercloud service could leverage other clouds to improve on the overall consistency. Finally, integrity of data and computation seems like a strong candidate for BFT in the Intercloud and, and as we already discussed, we are already witnessing research efforts in this direction [4, 6]. Clearly, this is not an exhaustive list and we expect to witness an increasing number of ideas for BFT applications in the Intercloud in the near future.

4 Conclusions

In this paper we summarized the limitations of Byzantine fault-tolerance for applications in the large-scale distributed systems of the inner-cloud. We also argued that Byzantine fault-tolerance is more suitable to the Intercloud, a cloud of clouds, which is emerging as the second layer in the cloud computing stack to complement the inner-cloud layer. The Intercloud offers an unprecedented failure independence at a low maintenance cost — this is provided by the diversity of cloud providers and differences in their internal implementations. Moreover, Byzantine failure model appears well suited for the threats that the Intercloud faces.

Acknowledgments. I would like to thank Christian Cachin, Robert Haas and Rachid Guerraoui for their valuable comments and many interesting discussions.

References

[1] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon. Racs: a case for cloud storage diversity. In *SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing*, pages 229–240, New York, NY, USA, 2010. ACM.

[2] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR fault tolerance for cooperative services. In *SOSP '05: Proceedings of the ACM SIGOPS 20th Symposium on Operating Systems Principles*, pages 45–58, New York, NY, USA, 2005. ACM.

[3] K. Birman, G. Chockler, and R. van Renesse. Toward a cloud computing research agenda. *SIGACT News*, 40(2):68–80, 2009.

[4] K. D. Bowers, A. Juels, and A. Oprea. HAIL: A high-availability and integrity layer for cloud storage. In *CCS '09: The 16th ACM Conference on Computer and Communications Security*, pages 187–198, 2009.

[5] C. Cachin. State machine replication with Byzantine faults. In Bernadette Charron-Bost, Fernando Pedone, and André Schiper, editors, Replication: Theory and Practice, LNCS, vol. 5959, pages 169-184. Springer, 2010.

[6] C. Cachin, R. Haas, and M. Vukolić. Dependable storage in the Intercloud. Research Report RZ 3783, IBM Research, Aug. 2010.

[7] C. Cachin, I. Keidar, and A. Shraer. Trusting the cloud. *SIGACT News*, 40(2):81–86, 2009.

[8] M. Castro and B. Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

[9] T. D. Chandra, R. Griesemer, and J. Redstone. Paxos made live: an engineering perspective. In *PODC '07: Proceedings of the 26th annual ACM Symposium on Principles of Distributed Computing*, pages 398–407, New York, NY, USA, 2007. ACM.

[10] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2):1–26, 2008.

[11] G. Chockler, R. Guerraoui, I. Keidar, and M. Vukolić. Reliable distributed storage. *IEEE Computer*, 42(4):60–67, 2009.

[12] A. Clement, M. Kapritsos, S. Lee, Y. Wang, L. Alvisi, M. Dahlin, and T. Riche. Upright cluster services. In *SOSP '09: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, pages 277–290, New York, NY, USA, 2009. ACM.

[13] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *SOSP '07: Proceedings of the ACM SIGOPS 21st Symposium on Operating Systems Principles*, pages 205–220, New York, NY, USA, 2007. ACM.

[14] K. Driscoll, B. Hall, H. Sivencrona, and P. Zumsteg. Byzantine fault tolerance, from theory to reality. In *SAFECOMP '03: Proceedings of the 22nd International Conference on Computer Safety, Reliability, and Security*, pages 235–248, 2003.

[15] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. In *SOSP '03: Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 193–206, New York, NY, USA, 2003. ACM.

[16] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *SOSP '03: Proceedings of the ACM SIGOPS 19th Symposium on Operating Systems Principles*, pages 29–43, New York, NY, USA, 2003. ACM.

[17] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić. The next 700 BFT protocols. In *Eurosys '10: Proceedings of the 5th ACM SIGOPS/EuroSys European Conference on Computer Systems*, pages 363–376, 2010.

[18] R. Guerraoui and M. Yabandeh. Independent faults in the cloud. In *LADIS '10: Proceedings of the 4th ACM SIGOPS/SIGACT Workshop on Large-Scale Distributed Systems and Middleware*, pages 12–16, 2010.

[19] J. Hendricks, G. R. Ganger, and M. K. Reiter. Low-overhead Byzantine fault-tolerant storage. In *SOSP '07: Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*, pages 73–86, New York, NY, USA, 2007. ACM.

[20] F. P. Junqueira and B. Reed. The life and times of a zookeeper. In *PODC '09: Proceedings of the 28th annual ACM Symposium on Principles of Distributed Computing*, page 4, 2009.

[21] P. Kuznetsov and R. Rodrigues. BFTW^3: Why? When? Where? Workshop on the theory and practice of Byzantine fault tolerance. *SIGACT News*, 40(4):82–86, 2009.

[22] L. Lamport. Paxos made simple. *ACM SIGACT News*, 32(4):18–25, 2001.

[23] L. Lamport. Lower bounds for asynchronous consensus. *FuDiCo '03: Future directions in distributed computing*, pages 22–23, 2003.

[24] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[25] D. Levin, J. R. Douceur, J. R. Lorch, and T. Moscibroda. TrInc: Small trusted hardware for large distributed systems. In *NSDI '09: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, pages 1–14, 2009.

[26] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[27] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *CCS '09: Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 199–212, New York, NY, USA, 2009. ACM.

[28] R. Rodrigues, M. Castro, and B. Liskov. BASE: using abstraction to improve fault tolerance. In *SOSP '01: Proceedings of the ACM SIGOPS 18th Symposium on Operating Systems Principles*, 2001.

[29] N. Santos, K. P. Gummadi, and R. Rodrigues. Towards trusted cloud computing. In *HotCloud '09: The 1st USENIX Workshop on Hot Topics in Cloud Computing*, 2009.

[30] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990.

[31] F. B. Schneider and L. Zhou. Implementing trustworthy services using replicated state machines. In Bernadette Charron-Bost, Fernando Pedone, and André Schiper, editors, Replication: Theory and Practice, LNCS, vol. 5959, pages 151-167. Springer, 2010.

[32] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Veríssimo. Highly available intrusion-tolerant services with proactive-reactive recovery. *IEEE Trans. Parallel Distrib. Syst.*, 21(4):452–465, 2010.

[33] W. Vogels. Eventually consistent. *Commun. ACM*, 52(1):40–44, 2009.