BASICS: A Multi-Blockchain Approach for Securing VM Migration in Joint-Cloud Systems

Pedro Braconnot Velloso, David Cordova Morales, Thi Mai Trang Nguyen, Guy Pujolle

BASICS: A Multi-Blockchain Approach for Securing VM Migration in Joint-Cloud Systems

Pedro B. Velloso*, David Cordova Morales*, Thi Mai Trang Nguyen*, and Guy Pujolle*
* LIP6, Sorbonne Université, CNRS, Paris, France
Email: {pedro.velloso, david.cordova, thi-mai-trang.nguyen, guy.pujolle}@lip6.fr

***Abstract*—Virtual Machine (VM) migration presents several advantages for both cloud operators and cloud users. The benefits of VM migration are amplified in the context of Joint-clouds, in which multiple clouds owned by distinct operators exchange VMs. However, inter-cloud VM migrations might require the usage of the Internet, which creates the possibility of external threats and therefore, a risk to the VMs' integrity. Thus, we propose a novel approach to secure VM migration for Joint-clouds. We consider both local migration and inter-cloud migration. Our solution is based on a multi-blockchain system, in which each cloud provider participates in the maintenance of a consortium blockchain to store relevant information about the migrations. In addition, each cloud operator might choose to keep a local blockchain to secure its local VM migrations. Therefore, inter-blockchain communication plays an important role in our work.**

I. Introduction

Cloud computing is a consolidated technology in which cloud operators deploy a set of physical servers that offers computing resources to their clients. Hence, these computer resources are often provided as virtual machines. The main idea consists of dividing the resources of physical servers into virtual machines that can be dynamically allocated according to the demand. Besides this basic concept, clouds from different operators might cooperate not only to scale resources for handling short-term spikes but also to provide adequate responsiveness and usability to clients distributed worldwide [1]. Hence, several solutions have been proposed to deploy cross-cloud cooperative architectures, for instance, Federated clouds [2], Joint-cloud [3], and Edge federation [4].

In this context, the migration of virtual machines plays a key role in bringing even more flexibility to cloud systems. VM migration allows cloud operators to perform server maintenance, load balancing, energy management, reduce service delay, avoid SLA violations, and provide security. Despite all its benefits, it is clear that VM migration introduces overheads to all the involved roles. Thus, several works try to improve the migration performance [5]. In addition, there are other issues to consider at the time of VM migration, such as respecting the SLA of the original VM [6], how to monitor SLA violations by different clouds [7], ensuring the same configuration in the new server, how to trust in VM migrations [8], or preventing infected VM migrations. This last issue is a challenge because one must keep the VM safe from attacks while it is running and assure the VM is not modified before arriving at the destination. During migration, a VM is vulnerable to DDoS attacks, loss of data integrity, confidentiality, unauthorized access, and virus contamination [9].

Hence, the main goal of this paper is twofold: (i) assuring the integrity of virtual machines during migration and (ii) keeping immutable records of VM migrations. Hence, our work is not concerned with protecting nor detecting VM attacks or malfunctioning. Instead, we aim to ensure that a VM has not been tempered after leaving its original server by taking a snapshot of its ultimate status using a one-way cryptographic function. In addition, upon the detection of a problem with a VM, cloud operators will be able to track down all the servers that have hosted this VM following all the migration history.

In this paper, we propose to benefit from the advantages of blockchain technology, specifically the immutability and high availability to achieve our goals. We consider a Joint-cloud scenario, in which several clouds from different tenants might exist. Virtual machines might migrate locally, i.e., inside their own cloud, or to an external one. Therefore, we propose a multi-blockchain system named BASIC, where each cloud provider participates in the maintenance of a consortium blockchain to record information about inter-cloud migrations. Additionally, each cloud provider might choose to keep a local blockchain to secure intra-cloud migrations. Inter-blockchain communication is possible via a node, which has access to both blockchain and functions as a gateway. Results show the impact of different parameters on the system performance, especially the influence of a generic consensus algorithm on the migration delay.

This paper is organized as follows. Section II presents the related work. Section III describes the architecture and the main characteristics of our multi-chain system. In Section IV, we detail our simulation scenario and parameters. In Section V, we present and analyze the simulation results. Section VI presents our conclusion and future work.

II. Related works

Most common approaches concentrate on mitigating specific attacks on virtual machines and their components [10] and to build trust on VM migration, especially when considering Joint-cloud scenarios [8], [11]. However, our goal does not focus on protecting nor detecting VM attacks or malfunctioning. Instead, we concentrate our efforts to verify the integrity of a VM when it arrives at the destination server.

In the most similar work, the authors [12] propose to secure VM migration using blockchain technologies. The authors consider only intra-cloud migration and use a unique blockchain to store every single VM migration procedure, by each physical server. Therefore, our approach differs in two aspects. First, we consider both intra- and inter-cloud migrations, which has led us to propose a multi-chain system. Second, to improve the system performance and, as a consequence, its scalability, we reduce the number of transactions that correspond to a unique migration. In another related work, the authors propose using a blockchain approach to secure the migration of Virtual Network Functions (VNF) [13]. Their goal is to store VNF configuration states in transactions, using a single blockchain. However, they do not consider a multi-cloud scenario.

Some work applies blockchain technology in a Joint-cloud context but does not consider VM migrations nor contemplate the possibility of multiple blockchains. Kai Wang *et al.* propose using blockchain technology to coordinate the collaborative services of vehicular Joint-cloud [14]. In [15], the authors use smart contracts to store the rights of users in Joint-cloud scenarios.

Several approaches consider multi-blockchain systems, for a variety of applications and thus, the concept of cross-chain communication arises. Hence, different cross-chain communications might exist with distinct complexity [16]. In general, there are three main approaches to cross-chain communications: (i) using an inter-chain protocol that is responsible for exchanging transactions among different chains [17]; (ii) using an intermediate chain combined with smart contracts [18], and (iii) a hybrid approach that considers the two previous ones [17].

In this paper, we propose a hybrid cross-chain approach in which a global blockchain is maintained by a consortium of cloud providers to ensure the integrity of VM inter-cloud migrations. Moreover, an inter-blockchain communication system is proposed by using nodes capable of routing transactions in both blockchains when using a local blockchain.

III. BASICS

In our scenario, several cloud operators need to migrate virtual machines not only inside their clouds, namely a local VM migration, but also across clouds from different cloud operators, namely, an inter migration. However, these virtual machines might suffer from malfunctioning problems, might be compromised due to security attacks, or even be modified along the path to the destination, like in a man-in-the-middle attack. In these cases, it is not recommended to accept such a machine in a new physical server, regardless of the type of migration (local or inter). Therefore, the main goal of this work is twofold: (i) preventing physical servers to accept a VM that was modified by third parties before its arrival; and (ii) providing means of tracking all VM migrations.

We assume that cloud operators are responsible for all migration decisions. Hence, after a migration decision is made, the cloud manager must follow four basic steps: (i) stop the execution of the virtual machine and (ii) generate unique identification (VM_{ID}), by applying a hash function to the VM content. Next, (iii) the cloud manager sends the VM to its new destination. Finally (iv), it creates a transaction, containing all relevant information about the VM migration, including the VM_{ID}, and sends it to the blockchain. Therefore, upon receiving the VM, the destination can apply a hash function to the content of the VM and then compare it to the VM_{ID} from the transaction associated with this migration, which is already in the blockchain. If the hash matches with the VM_{ID}, the migration has succeeded, otherwise the VM is discarded. This basic procedure prevents the destination server from accepting VMs that have been modified by third parties. In addition, it provides means of tracking the history of VM migrations.

A. System architecture

BASIC is a multi-chain approach in which each cloud operator participates in the maintenance of a consortium blockchain to store information on inter-cloud migration. The consortium blockchain has the advantage of decentralization and therefore, is trustless since cloud operators do not need to trust each other to reach a consensus on the state of the distributed ledger. Moreover, it provides high performance due to the limited and selective group of nodes participating in the blockchain. In addition, each cloud operator might choose to maintain a local blockchain to secure its intra-cloud migrations between the same or distant data centers. Participants of the local blockchain might be nodes operating in each data center of the cloud operator. Thus, the local blockchain is considered to be permissioned. In our case scenario, we have considered that each cloud operator maintains a local blockchain to evaluate the inter-chain communication. It is worth noticing, that the local blockchain and the consortium blockchain are independent and run their own consensus algorithm.

In this architecture, each cloud contains miners for the local blockchain, named *local validators*, as well as miners for the global chain, named *inter validators*. Cloud operators decide the amount of local and inter validators. Additionally, each cloud maintains a gateway node that receives the transactions from the cloud manager and is also responsible for the communication between the two blockchains.

B. Migration procedure

BASICS includes 4 types of transactions:

- **Indication**: this transaction indicates an attempt to migrate a VM to a specific physical server. It must include the VM_{ID}.
- **Confirmation**: the destination server use this transaction to confirm that the VM has been received and the hash matches the one included in the $Indication$ transaction.
- **Arrival**: it indicates that a new VM has been accepted in the cloud.
- **Leave**: it indicates that a VM from this cloud has successfully migrated to another cloud, and thus, no longer belongs to this cloud.

In step three from the basic procedure, after creating the transaction associated with the current migration, the cloud manager sends it to the cloud gateway.

Upon receiving a transaction, the cloud gateway verifies the type of migration and sends an $indication$ transaction to the proper set of validators, *local* or *inter*. Validators will mine blocks with a specific number of transactions. When the block is in the blockchain, one of the validators, in each cloud, announces to the cloud gateway all the new committed transactions in the current block that concerns its own cloud. Hence, whenever a new indication transaction is committed, the destination cloud gateway will receive it and send it to the destination cloud manager. Next, the hash verification occurs, and the cloud manager confirms the successful reception of the VM to the cloud gateway. The gateway creates a $confirmation$ transaction and sends it to all validators, according to the type of migration.

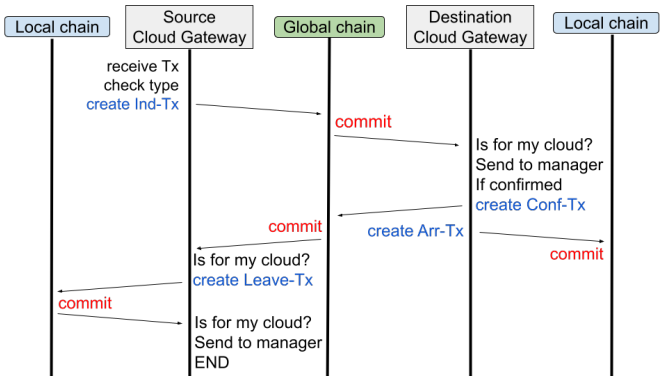


Fig. 1. BASICS - Inter-migration procedure

Similarly, the source cloud gateway will receive the confirmation, as soon the $confirmation$ transaction appears in a committed block. Therefore, whenever a cloud gateway receives a $confirmation$ transaction, it must verify whether it is a *local* or *inter* migration. In the first case, the migration has successfully finished, and it is already stored in the local blockchain because the VM has not left its cloud. However, if the $confirmation$ refers to an inter migration, then it means that the transaction has appeared in the global chain. Thus, the source gateway creates a $leave$ transaction in its local chain, and the destination gateway creates an $arrival$ transaction in its local chain. Hence, the inter migration can be considered finished only when a $leave$ transaction is committed in the local blockchain in the source cloud, as illustrated in Figure 1.

IV. Simulations

We use NS-3 to perform our simulation. To that, we have developed a blockchain and a generic consensus module that allows us to measure the performances of our cross-blockchain system based on a full-stack TCP/IP network infrastructure. Figure 2 shows the main scenario with three clouds where each cloud has a set of local and inter validators. The number of clouds and the number of validators are a parameter in our simulation. All nodes in a cloud are connected to a switch to simulate realistic cloud throughput, delays, and jitters. Besides, every cloud owns a router that is connected to the Internet.

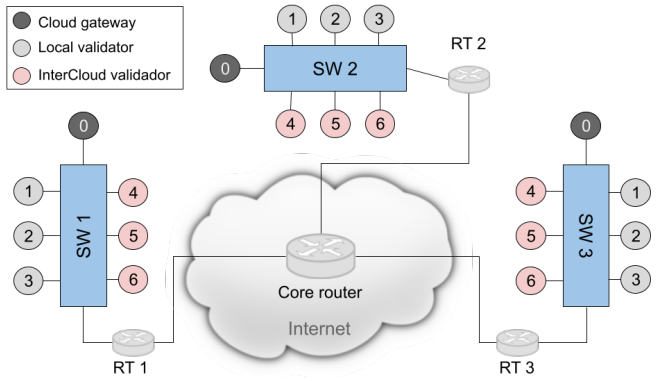


Fig. 2. Simulation scenario

The Internet is simulated by a single router that is connected via a point-to-point connection to every cloud router. Therefore, local validators are connected by a switch in a local network, while inter validators connect to their peers through a P2P overlay connection that traverses the Internet. Hence, we have two distinct communication delays for each blockchain. The validators implement the Blockchain module that manages the blockchain, while the gateways implement a TransactionApp module that creates and manages transactions.

A. Consensus

Since our multi-chain system supports different types of consensus, we implement a generic consensus that is aware of everything. The main idea consists of simulating different types of consensus with specific characteristics, just by tuning the parameters of the Oracle consensus. Hence, Oracle have three main parameters: $sendblockdelay$, $block-commitdelay$, and the $factor$. The block-commit delay corresponds to the time Oracle takes to send the block to all its peers. Therefore, considering a blockchain with np peers, with a block size of bck_{size}, measured in number of transactions, and the average size of transactions tx_{size}, the total delay for sending the block to all peers, considering multiple parallel connections is approximately given by:

$$delay\_send = np \times (D_{tr} + D_{pg}) + D_{qu} + D_{pc} \quad (1)$$

where D_{tr} is the transmission delay, D_{pg} is the propagation delay, D_{qu} is the queuing delay, and D_{pc} is the processing delay. We consider this the delay for committing a block after it was sent to all peers, as the time to process and receive the ACKs. As a simplification, we considered the same value as the time to send the block. As a consequence, we consider the total delay to commit the block is $commit\_delay_{min} = 2 \times Delay\_send$.

Finally, we assume that this commit_delay is the ideal consensus, namely a lower bound to the block-commit delay. To evaluate the impact of different consensus, we include the $Oraclefactor$ (O_f) parameter, which allows us to consider

different values for the block-commit delay, according to the minimum delay, explained above. Therefore, the block-commit delay is given by:

$$bck\_commit\_delay = O_f \times commit\_delay_{min}, \quad (2)$$

with $O_f \geq 1$. Besides, Oracle elects one leader, from the set of validators, responsible for generating the blocks in its blockchain. Table I summarizes the main parameters of our simulator along with their default values used for every simulation run.

Module	Parameters	Value
Topology	Number of clouds	4
	Number of Local validators	4
	Number of Inter validators	12
	CSMA data rate	10 Gbps
	P2P data rate	1 Gbps
	P2P Queuing delay	50 ms
Transactions App.	Avg. local mig. rate	1
	Avg. inter mig. rate	0.25
	Tx process delay	0.1 ms
Blockchain	Block size (txs)	16
Oracle	Oracle factor	1
	Election delay	100 ms

TABLE I
MAIN PARAMETERS AND STANDARD VALUES

V. RESULTS

To evaluate the performance of BASICS we run different sets of simulations and measure four performance metrics:

- **VM migration delay:** is defined by the time a migration transaction arrives at the gateway until the last transaction associated to this migration appears in one block in the local blockchain.
- **Block-commit delay:** is defined as the time interval between the moment the Blockchain module sends a new block B_i to the Oracle consensus and the Blockchain module adds the block B_i in the blockchain. This metric is expressed in Equation 2.
- **Inter-block delay:** is defined as the time interval between the inclusion of two consecutive blocks in the blockchain.
- **Number of blocks:** is defined as the number of blocks committed by the consensus and stored in the blockchain.

In each set of simulations, we vary different parameters to assess their impact on the system performance. Hence, for each simulation, we define the varying parameters and their respective values as well as specific values for particular parameters. All other parameters not mentioned assume the default values presented in Table I. The simulation time is 200 seconds. All results correspond to the mean of at least ten simulation runs, with a 95% confidence interval, represented by the error bars.

A. System stability

First, it is important to understand that our multi-chain system might present two basic behaviors regarding the migration delay, as shown in Figure 3, which illustrates the inter-migration delay variation over time in a simulation of 60 seconds, for two different migration rates. The blue line, with the lower rate, shows a bounded migration delay, and thus, a stable behaviour. The black line indicates the average delay of the blue line. However, the red line, with a much higher rate, manifests an unstable behavior with an unbounded delay. Therefore, it is crucial to respect the system capacity in terms of the migration rate to avoid unstable behaviours.

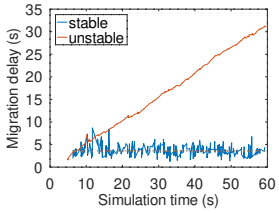
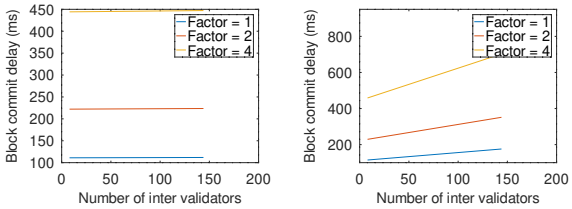


Fig. 3. Multi-chain stability

B. Oracle consensus performance

We evaluate the performance of Oracle regarding the block-commit delay. The main idea is to assess the impact of other parameters on the block-commit delay. Figure 4 reveals the influence of the number of inter validators on the block-commit delay, defined in Eq. 2, varying the oracle factor and the link data rate for the P2P connections. It is interesting to notice that the number of inter validators has no perceptive impact on the block-commit delay with a higher link data rate, even when we multiply by four the minimum commit delay, as shown in Figure 4(a). It is important to remember that we considered the possibility of creating multiple TCP connections. On the other hand, Figure 4(b) shows that, with a lower link data rate, increasing the number of validators augments the block-commit delay. Most important, Figure 4 illustrates how we can parameterize the oracle consensus to simulate different block-commit delays.



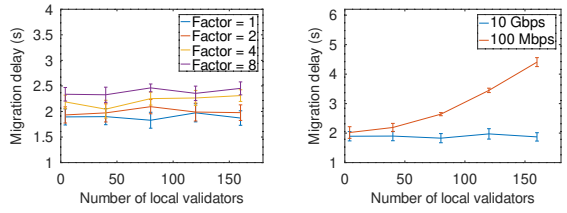
(a) P2P data rate = 1 GBps (b) P2P data rate = 10 MBps

Fig. 4. Block-commit delay

C. Local blockchain

To evaluate the local blockchain we isolated it from the global chain by setting the inter-migration rate to zero. Hence,

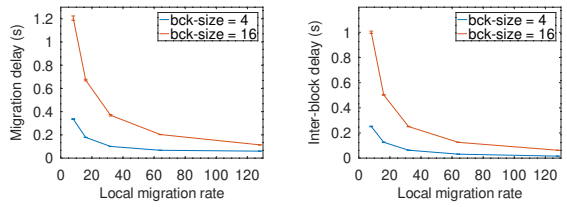
there are only local migrations. First, we analyze the effect of the number of local validators on the migration delay. Figure 5 shows the delay when we vary the number of validators from 4 to 164. In Figure 5(a), we use different values for the block-commit delay, represented by the Oracle factor. Results show that the variation of the number of validators has no perceptive impact on the migration delay. However, the block-commit delay directly affects the migration delay. In Figure 5(b), we use two different values for the data link rate of the local network. We observe that increasing the number of validators with a lower data link rate has a significant impact on the migration delay.



(a) CSMA data link rate = 10 Gbps (b) Oracle factor = 1

Fig. 5. The effect of the number of local validators on the migration delay

Next, we vary the local migration rate to evaluate the migration delay and the scalability of the local blockchain regarding the number of migrations per second. Figure 6 shows the migration delay with different migration rates, using two block sizes. First, we can observe that the size of the block is an important parameter to define the migration delay, as expected. The larger is the size of the block, the longer the system will take to fill the block, considering a specific migration rate. Therefore, the inter-block interval increases as shown in Figure 6(b). It is important to remember that increasing indefinitely the migration rate is not an option since, at some point, the system reaches its own limit beyond which the stability of the system is compromised. In this specific scenario, the limit is between 128 and 256 migrations per second.

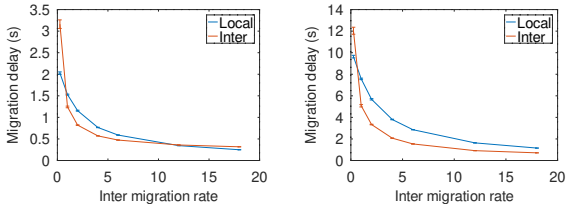


(a) Migration delay (b) Inter-block delay

Fig. 6. Local migration with different migration rates for block sizes of 4 and 16 transactions.

D. Global blockchain

We evaluate the behavior of the global chain regarding the inter-migration delay. Figure 8 shows the results for the inter-migration delay when varying the inter-migration rate from 0.25 to 16 migrations per second for two block sizes. We can observe that, similar to the local blockchain, increasing the inter-migration rate allows the leader to complete a block faster, and as a consequence, we can reduce the inter-migration delay significantly. However, for the global chain, the limit on the migration rate is much lower than the local chain, due to the significant difference in the delay for committing a block in both blockchains. The main reason for this discrepancy is the difference in the Round Trip Time (RTT), since local migrations are confined to a local network, while inter migrations must traverse the Internet.

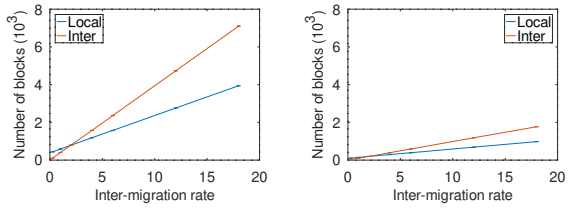


(a) Block size = 4 transactions (b) Block size = 16 transactions

Fig. 7. The effect inter-migration rate on the migration delay with different block sizes

Another interesting result is the impact of the inter-migration rate on the local blockchain. The blue curves in Figure 8 indicate that increasing the inter-migration rate reduces the local migration delay. Indeed, inter migration also generates local transactions, as explained in Section III-B.

Figures 8(a) and 8(b) point out the total number of blocks in the local chains and the global chain. Results confirm that using a larger block size implies fewer blocks, as expected. In addition, we observe the effect of the inter migrations generating local transactions in the local chains. Thus, the rise in the number of blocks in the local chain is entirely due to the inter migrations since the number of local migrations does not change.



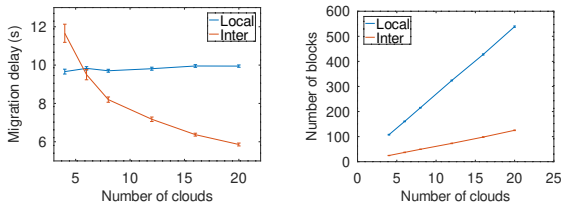
(a) Block size = 4 transactions (b) Block size = 16 transactions

Fig. 8. The effect inter-migration rate on the number of blocks

E. Multi-chain system

After understanding the performance of the two blockchains separately, it is important to evaluate the whole multi-chain system. First, we assess the impact of the number of participating clouds on the system performance. To accomplish this goal, we fixed the number of validators and migration rates to their default values and vary the number of clouds from 4 to 20. Figure 9(a) shows that increasing the number of clouds

only affects the global chain performance. This is an expected result, since every cloud generates a certain number of local and inter migration. Hence, local migrations are confined to each cloud, and as a consequence, the number of clouds has no effect on the local-migration delay. However, as inter migrations share the same global blockchain, augmenting the number of clouds leads to a rise in the overall migration rate, which reduces the migration delay, as discussed previously. Even though inter migrations also generate transactions in the local blockchains, the increase in the overall inter-migration rate does not affect the performance of local chains because these local transactions are proportionally spread among the participating clouds.

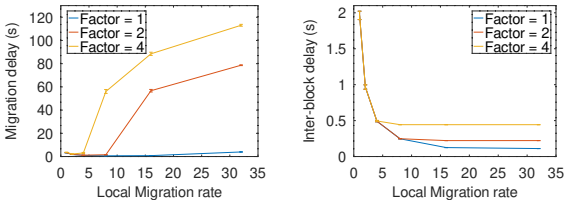


(a) Migration delay (b) Number of blocks

Fig. 9. The effect of the number of clouds on the performance of our multi-chain system.

Figure 9(b) shows the number of blocks in the local chains and global chain when varying the number of clouds. Results confirm that increasing the number of clouds augments the number of blocks in the global and local blockchains. Besides, the blue curve concerning the local chains presents a higher slope. It occurs because, additionally to the rise of the number of blocks due to the augmentation of the number of clouds, there is the effect of the inter migrations on the local chains, as discussed above.

Afterward, we set the proportion between local and inter migrations rates to 4:1, in each cloud. Then, we vary the local-migration rate from 1 to 32 for different block-commit delays (Oracle factors). Figures 10 illustrate the results for the global chain with a block size of 4. We can observe the shift in the stability point when the block-commit delay increases. Figure 10(b) indicates when the inter-block delay reaches the minimum block-commit delay that corresponds to the point beyond which the system becomes unstable.



(a) Inter-migration delay (b) Inter-block interval

Fig. 10. The effect migration rate on the inter migration delay and inter-block delay, with 4:1 proportion and different Oracle commit delays.

VI. Conclusion

In this paper, we propose BASICS, a multi-blockchain system to secure VM migration in Joint-cloud systems. We implement the proposed solution in NS-3 to evaluate the system performance and its scalability. We developed emulate a generic consensus, named Oracle, that emulate consensus with different characteristics. Hence, we evaluate our system in terms of migration delay, inter-block delay, the number of blocks, and the block commit-delay. We assess the influence of several parameters on the performance of our multi-chain systems. Results allow us to characterize and understand the behavior of the system as well as its relations to the main parameters. They also show the stability limitation of our systems and the trade-off between increasing the system capacity and reducing the average migration delay.

References

[1] N. Grozev and R. Buyya, "Inter-cloud architectures and application brokering: taxonomy and survey," *Software Practice Experience*, 2012.

[2] C. A. Lee, "Cloud federation management and beyond: Requirements, relevant standards, and gaps," *IEEE Cloud Comp.*, vol. 3, no. 1, 2016.

[3] P. Shi, H. Liu, S. Yang, Y. Zhang, and Y. Zhong, "The inherent mechanism and a case study of the constructional evolution of the JointCloud ecosystem," *IEEE Internet of Things J.*, vol. 7, no. 3, 2020.

[4] X. Cao, G. Tang, D. Guo, Y. Li, and W. Zhang, "Edge federation: Towards an integrated service provisioning model," *IEEE/ACM Transaction on Networking*, 2020.

[5] M. Noshy, A. Ibrahim, and H. A. Ali, "Optimization of live virtual machine migration in cloud computing: A survey and future directions," *Journal of Network and Computer App.*, vol. 110, pp. 1–10, 2018.

[6] W. A. Ghumman and A. Schill, "Continuous and distributed monitoring of cloud slas using s3lacc," in *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, 2017.

[7] A. A. Falasi, M. A. Serhani, and R. Dssouli, "A model for multi-level SLA monitoring in federated cloud environment," in *IEEE UIC/ATC'13*, 2013.

[8] U. Ahmed, I. Raza, and S. A. Hussain, "Trust evaluation in cross-cloud federation: Survey and requirement analysis," *ACM Computing Surveys*, vol. 52, no. 1, 2019.

[9] K. Janjua and W. Ali, "Enhanced secure mechanism for virtual machine migration in clouds," in *International Conference on Frontiers of Information Technology (FIT)*, 2018.

[10] R. Patil and C. Modi, "An exhaustive survey on security concerns and solutions at different components of virtualization," *ACM Computing Surveys*, vol. 52, no. 1, 2019.

[11] M. Aslam, S. Bouget, and S. Raza, "Security and trust preserving inter- and intra-cloud VM migrations," *International Journal of Network Management*, vol. 31, no. 2, p. e2103, 2021.

[12] N. Bozic, G. Pujolle, and S. Secci, "Securing virtual machine orchestration with blockchains," in *CSNet'17*, 2017.

[13] I. D. Alvarenga, G. A. F. Rebello, and O. C. M. B. Duarte, "Securing configuration management and migration of virtual network functions using blockchain," in *IEEE/IFIP NOMS'18*, 2018.

[14] B. Yin, L. Mei, Z. Jiang, and K. Wang, "Joint cloud collaboration mechanism between vehicle clouds based on blockchain," in *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, 2019.

[15] P. Zheng, Z. Zheng, W. Chen, J. Bian, and J. E. Yang, "Ethershare: Share information in jointcloud environment using blockchain-based smart contracts," in *IEEE SOSE'19*, 2019.

[16] I. A. Qasse, M. A. Talib, and Q. Nasir, "Inter blockchain communication: A survey," in *ArabWIC 2019*, 2019.

[17] M. Borkowski, P. Frauenthaler, M. Sigwart, T. Hukkinen, O. Hladky, and S. Schulte, "Cross-blockchain technologies: Review, state of the art, and outlook," in *White paper*, 2019.

[18] Y. Jiang, C. Wang, Y. Wang, and L. Gao, "A cross-chain solution to integrating multiple blockchains for IoT data management," *Sensors*, vol. 19, no. 2042, 2019.