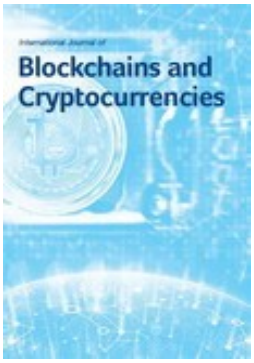
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Blockchains and Cryptocurrencies

ISSN online: 2516-6433 - ISSN print: 2516-6425

https://www.inderscience.com/ijbc

ChainElastic: a cloud computing resource elasticity model for IoT-based blockchain applications

Vinicius Facco Rodrigues, Josué Valtair Silva e Silva, Rodrigo Da Rosa Righi, Cristiano André Da Costa and Alex Roehrs

DOI: 10.1504/IJBC.2021.117806

Article History:

Received:	26 April 2020
Accepted:	20 August 2020
Published online:	25 September 2021

ChainElastic: a cloud computing resource elasticity model for IoT-based blockchain applications

Vinicius Facco Rodrigues*,
Josué Valtair Silva e Silva,
Rodrigo da Rosa Righi,
Cristiano André da Costa and
Alex Roehrs

Universidade do Vale do Rio dos Sinos (UNISINOS),
São Leopoldo, RS, Brazil
Email: vfrodrigues@unisinos.br
Email: josue.silva@outlook.com
Email: rrrighi@unisinos.br
Email: cac@unisinos.br
Email: alexr@unisinos.br
*Corresponding author

Abstract: Internet of things (IoT) environments are composed of a changing number of devices that produce data events at different rates. The development of blockchain solutions for such environments characterises an emerging trend of applications, thus requiring the system to be scalable. Nowadays, cloud computing technology became a standard solution to provide scalable environments for many types of applications through the cloud elasticity feature. However, when scale-in operations occur, cloud computing platforms operate by deleting virtual machine instances erasing all their data. In the blockchain scope, as each node has the entire transaction history stored, the data is lost, which requires new nodes to download all the history again before becoming operational. Consequently, those operations reflect a significant impact on the system's performance. In this context, this article proposes ChainElastic, a cloud elasticity model to run blockchain IoT applications that maintains the database history transactions from removed nodes. We developed a prototype that runs over the OpenNebula cloud, highlighting the benefits of using elasticity in terms of resource consumption, latency, and execution time. The experiments demonstrate gains of 39.64% in resource consumption with the ChainElastic model when comparing it against a scenario with fixed resources.

Keywords: blockchain; internet of things; IoT; cloud computing; cloud elasticity; ChainElastic.

Reference to this paper should be made as follows: Rodrigues, V.F., Silva e Silva, J.V., da Rosa Righi, R., da Costa, C.A. and Roehrs, A. (2021) 'ChainElastic: a cloud computing resource elasticity model for IoT-based blockchain applications', *Int. J. Blockchains and Cryptocurrencies*, Vol. 2, No. 1, pp.1–18.

Biographical notes: Vinicius Facco Rodrigues is a researcher at the Universidade do Vale do Rio dos Sinos (UNISINOS), Brazil. Currently, he is a PhD student from the Applied Computing Graduate Program (PPGCA) at UNISINOS. He obtained his BSc in Computer Science and his MS in Applied Computing from UNISINOS, in 2012 and 2016, respectively. His research interests include distributed systems, high performance computing, cloud computing, wireless sensor networks, and healthcare.

Josué Valtair Silva e Silva finished his BSc in Computer Science in 2018 focusing mainly in blockchain applications. His research interests also include distributed systems and cloud computing environments.

Rodrigo da Rosa Righi is a Professor and researcher at the University of Vale do Rio dos Sinos (UNISINOS), Brazil. He concluded his post-doctoral studies at the KAIST – Korean Advanced Institute of Science and Technology, South Korea. He obtained his MS and PhD in Computer Science from the Federal University of Rio Grande do Sul, Brazil, in 2005 and 2009, respectively. His research interests include performance analysis, scheduling, load balancing on cluster, grid and cloud environments.

Cristiano André da Costa is a Full Professor and researcher at the Universidade do Vale do Rio dos Sinos (UNISINOS), Brazil, working at the Applied Computing Graduate Program and directing the SOFTWARELAB, UNISINOS Laboratory of Software Innovation. He obtained his PhD and MSc in Computer Science from the Universidade Federal do Rio Grande do Sul (UFRGS), Brazil. Since 2012, he is a researcher of Productivity at the National Council for Scientific and Technological Development (CNPq, Brazil).

Alex Roehrs is a Professor of Computer Science and PhD student in Applied Computing at the Universidade do Vale do Rio dos Sinos (UNISINOS), Brazil. He also obtained his MS in Applied Computing from UNISINOS in 2012.

1 Introduction

Blockchain is an emerging technology based on a network-based consensus validation mechanism mostly used in the financial field (Morktexs et al., 2019; Lee et al., 2020; Furtado et al., 2020). Although it has gained notoriety by its use in the cryptocurrency scope, blockchain applications are not restricted to this area. In the last few years, strategies have been focusing on employing this new technology in the internet of things (IoT) field (Gao et al., 2018; Liu et al., 2018; Moin et al., 2019; Dorri et al., 2019; Wang et al., 2020) because of its capability of increasing data security. Private blockchain is an essential approach for industries and companies that seek security in the step of validating events and processes (Reyna et al., 2018; Seigneur et al., 2020). For instance, in assembly lines of supply chains and organisation processes, it is crucial to ensure that data (that describe events) are correct and not corrupted by illegal alterations. It is critical for audits that require data integrity to track what happened and the history of events.

In the scope of Industry 4.0, in which numerous sensors generate and coordinate data, stakeholders need to trust results (Bodkhe et al., 2020). Companies and industries rely entirely on the data so that managers can make decisions with the warranty of the integrity of the stored data. Approaches employing machine-to-machine (M2M) applications, where a particular perception in one machine (sensor-actuator) triggers activity on another device, must rely on the actions performed in the workflow (Johng et al., 2020). Thus, IoT-based blockchain is vital in M2M environments since it stores data related to cause-effect activity in a block (Johng et al., 2020; Killer et al., 2020). In such an environment, both (source and destination) or many (source and destinations) rely precisely on what happened.

Since blockchain operates as a decentralised database, the entire transaction history is stored entirely on each node (Lee et al., 2020). This decentralised storage nature is not suitable for cloud solutions that manage resources by adding or removing virtual machine (VM) instances. In general, cloud computing platforms operate by deleting these instances erasing all the data contained in it (Galante and de Bona, 2012). This feature becomes unwise for blockchain applications because, at the time of resource allocation, the new application instance needs to accomplish the consensus of the network and copy all the transactions carried out so far. Depending on the chain's size, this operation may take more than one day for a VM to be fully available to the user. To the best of the authors' knowledge, there are no solutions that try to combine performance, portability, and resource elasticity in the scope of blockchain applications.

In this context, this article presents ChainElastic, a private cloud elasticity model for IoT cloud-based blockchain applications. ChainElastic employs threshold-based cloud elasticity policies (Galante and de Bona, 2012; Kuhlenkamp et al., 2020; Zhong and Buyya, 2020) to offer automatic virtual resource management for private blockchain in cloud computing environments. This feature consists of reorganising VM instances that are running a blockchain environment according to the system load. We propose an elasticity manager that monitors VM instances running nodes of the blockchain network. It identifies increasing or decreasing system load situations and performs adjustments in the number of available resources.

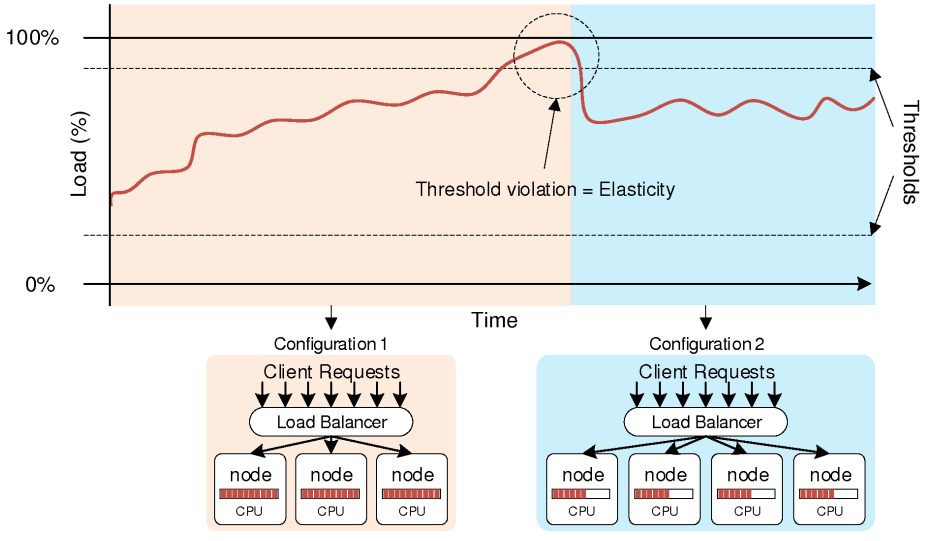
Figure 1 illustrates the proposed strategy showing a situation in which the increase in the load triggers a resource reorganisation from a configuration state (configuration 1) to another (configuration 2). The new configuration contains more VM instances allowing a better workload distribution, which decreases the overall system load. Through this strategy, ChainElastic can increase the system performance in high load situations, and improve resource efficiency when there is no need for high computational processing power. Also, ChainElastic presents a proposal to provide elasticity without changing blockchain characteristics, so being useful for different existing blockchain implementations. We developed a prototype that runs over OpenNebula (https://opennebula.org/) cloud middleware and MultiChain (https://www.multichain.com/) platform. The results were encouraging, with gains in efficiency and economy in the execution of malleable blockchain applications.

In summary, ChainElastic brings two contributions to the state-of-the-art in private blockchain applications:

1 An IoT-based private blockchain elasticity model for cloud computing environments with an auto organisation of VM instances.

2 An elasticity strategy that both preserves the state of application nodes, also not imposing the maintenance of computing resources of VM that are currently unused. Thus, VMs are frozen when unused and updated from this last state when activated again (in the case of a scale-out operation).

Figure 1 ChainElastic elasticity strategy (see online version for colours)



Notes: The system load violates the upper threshold causing the model to increase the number of resources.

2 Related work

The literature presents some studies that focus on performance improvement and scalability on blockchain models and applications. In general, strategies focus on developing solutions for cryptocurrency, mostly concentrated on providing alternatives for Bitcoin. In this section, we describe some related work in the blockchain context, and we divide it into two subsections. In Subsection 2.1, the state-of-the-art regarding blockchain solutions is detailed. Following, in Subsection 2.2, the analysis of the studies and motivation of the current article is presented.

2.1 State-of-the-art

Croman et al. (2016) conducted a study to demonstrate the challenges of scaling Bitcoin and blockchain. They stated two types of limitations, which are related to reparametrisation and infrastructure issues. Although reparametrisation improves scalability, this improvement is less than an adjustment in the network infrastructure. Also targeting cryptocurrency, Eyal et al. (2016) presented Bitcoin-NG, a blockchain that serialises transactions offering better latency and bandwidth than Bitcoin without changing the other properties. The performance gain of Bitcoin-NG over Bitcoin is related to the choice of the leading node. Because only a single leader node processes the

transactions, it avoids validation waste of the same operation by two nodes. Sompolinsky and Zohar (2015) proposed GHOST as an alternative protocol to Bitcoin that changes the most extended chain rule in case of a fork in the blockchain. For each fork in the string, GHOST selects the heavier sub-tree rooted in it. GHOST does not select the longest chain. Therefore, the growth rate is lower, making the algorithm achieve better performance, thus having a smaller loss in growth rate and having no influence on security.

Bruce (2014) proposed mini-blockchain, a new cryptocurrency similar to Bitcoin but with several different aspects. Mini-blockchain is the mechanism responsible for coordinating how the network processes transactions. Its main differential is that it does not require to keep the copy of the historical blocks. Rizun (2016) proposed Subchains, a weak block technique that provides incentives for miners to cooperate for the benefit of the network. Weak blocks are blocks that meet the work test's requirement, but their purpose is ineffective because it has a value higher than the difficulty goal. Its technique reduces block propagation time because only the newer transactions need to be propagated on the network, reducing the probability of orphaned blocks. Finally, Ozisik et al. (2017) introduced Graphene, a proposal to change the method of announcing new blocks. This proposal is suitable for various blockchain-based network protocols, thus encoding the blocks to the size of 2.6 kilobytes (KB), and the Bitcoin default block size of 1 megabyte (MB). Graphene uses an iterative combination of bloom filters and invertible bloom lookup tables (ITBL). This combination is useful to the set reconciliation problem in the P2P network.

Finally, Dorri et al. (2019) focus on the scalability limitations of blockchain, focusing on IoT environments. They propose a lightweight scalable blockchain (LSB) that employs a distributed time-based consensus algorithm. They divide the nodes into clusters and use a time division strategy in which each cluster head generates new blocks only in a custom consensus period.

2.2 Analysis and motivation

Table 1 summarises the initiatives highlighting their main characteristics. In addition to compatibility and elasticity issues, the table depicts five abstract planes according to the definition of Croman et al. (2016):

1 network: transaction propagation strategies

2 consensus rule: transactions acceptance operations

3 storage: global memory for authenticated data

4 view: support to local view of the ledger

5 side: off-chain functionalities.

The studies have at least one of the following limitations:

1 articles focusing on scalability problems do not detail the bottlenecks that cause such issues or do not present proposals to improve the blockchain performance

2 owing to Bitcoin be the main application of blockchain and it is not an academic proposal, many contents concerning blockchain are out of the scientific bases but on websites and in developing communities of blockchain topics.

We identified a gap regarding the allocation of resources because none of the works handle the dynamic allocation of resources to validate transactions. Therefore, in these systems, we can have either case of idle resources generating waste or examples of resources operating at maximum capacity, causing delays in the system. This gap becomes critical when addressing private blockchain because we can have under or over-system utilisation more commonly than public clouds. Moreover, the related work presents a limitation regarding compatibility, because, except Graphene, the exhibited works change the blockchain concept rendering incompatible with the proposal of existing blockchain applications profoundly.

Table 1 Related work comparison according to their main limitation target: (a) network (b) consensus rule (c) storage (d) view (e) side (f) compatibility (g) elasticity

Study	*Abstract planes*						
	(a)	*(b)*	*(c)*	*(d)*	*(e)*	*(f)*	*(g)*
Dorri et al. (2019)		✓					
Bitcoin-NG (Eyal et al., 2016)		✓					
GHOST (Sompolinsky and Zohar, 2015)		✓					
Mini-blockchain (Bruce, 2014)		✓	✓	✓			
Subchain (Rizun, 2016)		✓			✓		
Graphene (Ozisik et al., 2017)	✓					✓	

3 The ChainElastic model

In this section, we present the ChainElastic model to fill the previously explained gaps. ChainElastic is a model to provide dynamic resource allocation in the context of private blockchain. The next sections introduce the model in detail. In Subsection 3.1 the design decisions are presented. Next, in Subsection 3.2, the model architecture is described, and in Subsection 3.3, details of the ChainElastic manager are given. Following, in Subsection 3.4, the resource elasticity model on blockchain applications is detailed. Finally, in Subsection 3.6, two new performance and resource consumption metrics to assess the evaluation of the model are proposed.

3.1 Design decisions

ChainElastic provides an elasticity manager component designed for private cloud environments. As we are targeting private blockchain applications, we adopted private cloud infrastructures since they offer a secure and controlled environment (Zhang et al., 2010). In the context of IoT-based applications, cloud environments provide an easy way of sharing information from distributed sources (Chen et al., 2019). The ChainElastic elasticity manager performs scale-in and scale-out operations by activating and deactivating VM instances from the cloud resource pool. One of the main challenges

in cloud environments is to balance the trade-off between performance and cost (Mireslami et al., 2017). Through a novel strategy, by just disabling a particular VM, ChainElastic keeps its state without deleting it from the cloud. This strategy decreases the time delay of deploying a new VM instance, also reducing network resource waste for this operation.

On the other hand, when adding new resources, the resource manager turns on deactivated VM instances. This method accelerates the consensus process since the new instance already has a chunk of the operation history. The user must configure all VM instances in advance and provide them as input to the elasticity manager. The manager is in charge of all monitoring and decision making operations automatically. Any blockchain application can run in the environment, and there is no need to change management and monitoring rules. For better performance of the environment, the application should have a division policy to validate the transactions through the available nodes. Considering this background, we developed ChainElastic model with the following design decisions in mind:

1 Compatibility: The user does not need to change its application to use ChainElastic.

2 Automatic elasticity: The user defines the minimum and maximum thresholds regarding the number of resources. Therefore ChainElastic must be able to achieve the elasticity automatically afterward.

3 VMs: The user informs which are the VMs available with the blockchain and the ChainElastic elasticity manager controls only these VMs.

4 Private cloud: We are addressing private blockchain. Thus, we deploy the ChainElastic prototype in a private cloud infrastructure.

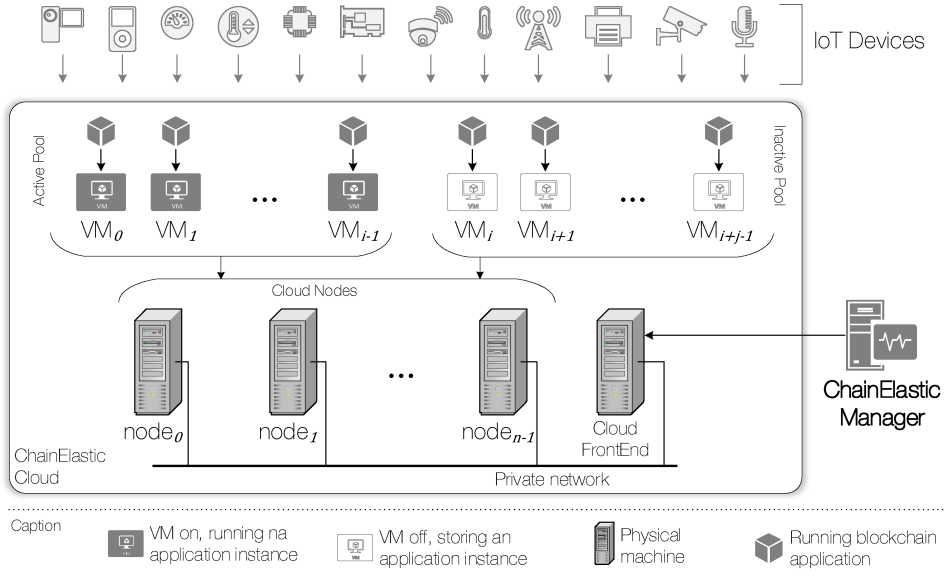
3.2 Architecture

Figure 2 depicts ChainElastic's architecture. The main component is the ChainElastic manager, responsible for monitoring the system load and performing elasticity operations. These operations consist of enabling or disabling VM instances applying a threshold-based rule algorithm. Users must first create the VM instances and deploy their blockchain application in them through the Cloud FrontEnd. Additionally, the user must inform all available VMs in the cloud to the ChainElastic manager. An SLA-based file (Wu et al., 2015) does this process defining the VM instances list, and the minimum and maximum limits for active instances.

A total of n physical machines compose the cloud infrastructure, and they operate as computing nodes that run VM instances. ChainElastic uses c VM instances per node, in which c is the number of cores available in a specific node. This approach aims at improving the VMs performance, based on the work of Lee et al. (2011). Hence, it allows the cloud infrastructure to be composed of heterogeneous physical nodes. The only requirement is the number of VMs matches the number of available cores of all physical nodes. Additionally, a physical machine acts as the Cloud FrontEnd performing management and monitoring tasks. More importantly, the FrontEnd provides an application user interface (API), allowing the manager to gather information from VM instances and request resource reorganisation operations. The manager has

the flexibility of running either inside or outside the cloud environment. The only requirement is that the manager should be able to reach the FrontEnd through the API.

Figure 2 ChainElastic architecture



Notes: The model comprises i online VM instances, j offline VM instances, and n physical nodes. The ChainElastic manager controls the virtual resources through the cloud provider API provided by the FrontEnd server.

The architecture is composed of i online (active) and j (inactive) offline VMs, which are divided into two different pools according to their state:

1 active pool, which comprehends online VMs

2 inactive pool, which contains offline VMs.

Online VM instances are actively running the blockchain application. These instances run in a physical node consuming hardware resources. The monitoring process considers only them to compute the system load because they are the only ones executing the blockchain environment.

On the other hand, offline VM instances are inactive, and, as they are not running the blockchain application, they are not consuming hardware resources. The physical node in which the VM is deployed holds the VM disk image stored. Hence, when the elasticity process activates it, the Cloud FrontEnd does not need to transfer the VM image over the network again. It only needs to turn its operating system (OS) on, and the boot process starts automatically. It avoids wasting time transferring a new image from the cloud FrontEnd to the physical node. Additionally, this strategy also accelerates the consensus process because offline VM instances have their data preserved even after the shutdown process. Therefore, once this VM is activated again, the blockchain application already has a chunk of the chain operations history. Hence, it only needs to

download a portion of data from the network. This strategy avoids a long waiting time for the consensus process finish, and the node can perform operations in the network earlier.

3.3 ChainElastic manager

The ChainElastic manager performs monitoring observations in a periodic manner. Each monitoring event consists of acquiring the CPU values from each online VM and computing the system load for decision-making. This process uses measures only from online VM instances since they are consuming cloud resources. We opted by the CPU metric because the blockchain consensus algorithm is CPU-intensive, and auto-scaling strategies commonly use this metric (Al-Haidari et al., 2013). At each monitoring cycle, the manager performs elasticity operations according to the system load. These operations consist of changing the state of VM instances by turning them on or off if a specific threshold is violated (see detail in Subsection 3.4). All elasticity actions are performed automatically and do not require user intervention.

The manager has a policy of fault tolerance by checking at each monitoring cycle if there is at least one online VM instance. If not, it allocates one from the inactive pool by turning it on. Therefore, the VM initialises the OS and automatically starts the user application. At the end of this process, the manager changes the VM to the active pool and returns to the regular monitoring process. The communication between the blockchain application on different VM instances, transaction replication, and consensus policy, are the responsibility of the blockchain middleware and its peer-to-peer network.

Aiming at preventing consecutive elasticity operations at very short processing peaks, the manager has a value to address the cool-down period. The cool-down defines the number of monitoring cycles that the manager should wait to perform new elasticity reorganisations. Through this strategy, ChainElastic ensures that elasticity operations will not be executed while the blockchain model performs redistribution of transaction validations. Further, the adoption of this strategy is pertinent to avoid the Hysteresis problem (Mayergoyz, 2003), with consecutive elasticity actions.

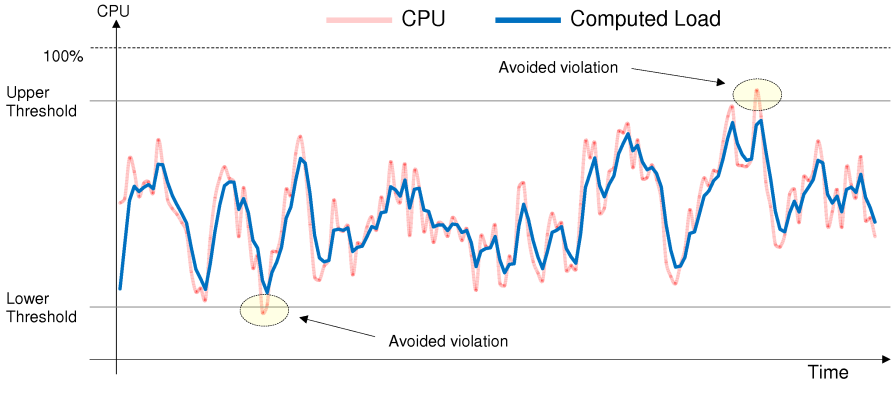
3.4 Elasticity model

ChainElastic employs a modified version of the AutoElastic elasticity model (Righi et al., 2016). AutoElastic is a threshold-based elasticity model for high-performance computing (HPC) applications in the cloud. Similarly, ChainElastic uses a threshold-based strategy to provide reactive and horizontal elasticity. At each monitoring cycle, the manager computes the metric $Load_k$ [equation (1)] for each online VM k, and then the metric $Load$ [equation (2)] to assess the system load.

According to equation (1), let o be the monitoring cycle index, $k \in \{0, 1, ..., i-1\}$ representing the index of one of the i online VM instances, and $CPU_k(o)$ the CPU load of the VM k in the monitoring observation o. Then $Load_k$ computes the load of the VM k in the monitoring observation o. $Load_k$ employs a simple exponential smoothing technique (Herbst et al., 2013) in which all measure observations receive a weight according to their age (the newer the value, the higher the weight). This strategy avoids either false-negative or false-positive elasticity actions in peak situations, as illustrated in Figure 3.

$$Load_k(o) = \begin{cases} \frac{CPU_k(o)}{2} & \text{if } o = 0 \\ \frac{Load_k(o-1)}{2} + \frac{CPU_k(o)}{2} & \text{if } o \neq 0 \end{cases} \quad (1)$$

Figure 3 ChainElastic threshold-based elasticity model employing a simple exponential smoothing in CPU measures (see online version for colours)



Notes: The figure highlights two specific points in which the strategy avoids unnecessary operations.

Finally, equation (2) defines how the manager computes the $Load$ metric of the system in the o^{th} monitoring observation. The equation computes the arithmetic mean of the load of all online VM instances.

$$Load(o) = \frac{\sum_{k=0}^{i-1} Load_k(o)}{i} \quad (2)$$

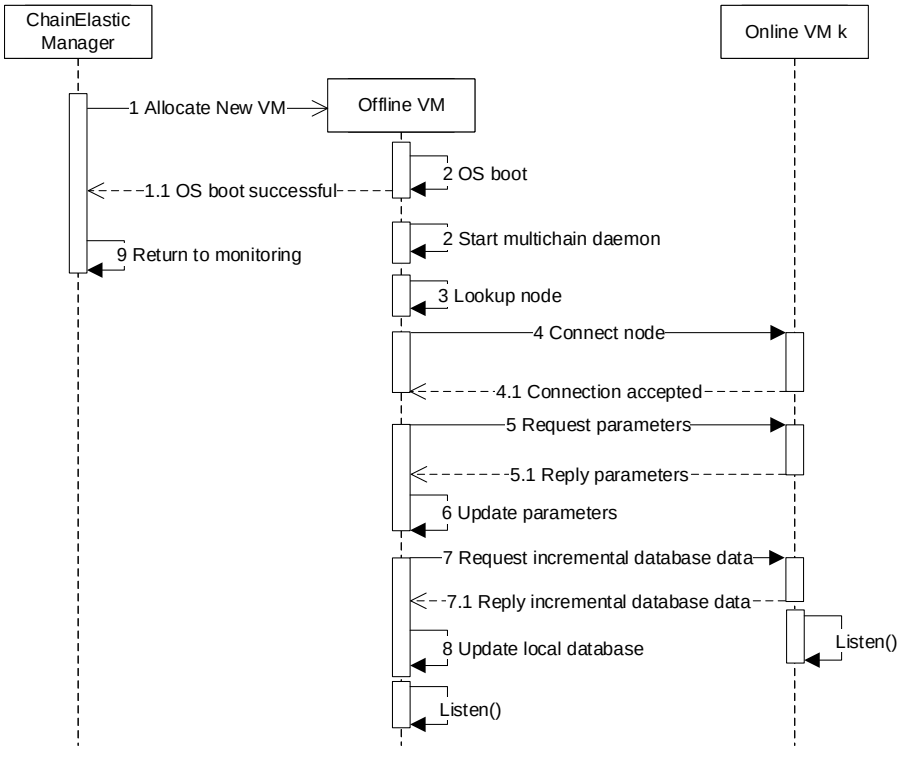
Based on the $Load$ metric, an upper ($UpperT$) and a lower ($LowerT$) threshold, elasticity operations take place if one of the following conditions is satisfied: (C1) $Load > UpperT$; or (C2) $Load < LowerT$. If condition C1 is $true$, then the manager increases the number of online VM instances (scale-out) by turning on one VM instance from the offline VM pool. On the other hand, if condition C2 is $true$, then the manager decreases the number of online VM instances (scale-in) by turning off one VM instance from the online VM pool. If none of the conditions is satisfied, the manager does not take any actions and waits for the next monitoring cycle. The values for $UpperT$ and $LowerT$ are fixed parameters fulfilled by the user when starting the manager. In other words, the values entered by the user for thresholds remain the same until the manager is stopped.

There is no communication between ChainElastic and the blockchain application when increasing or reducing resources. Thus, the middleware's network protocol identifies the nodes available in the network and the transaction distribution policy. After an elasticity operation, the manager does not perform any further actions, respecting the cool-down observation period. During such a period, even if $Load$ violates a threshold, elasticity actions do not occur.

3.5 Application model

The application model is based on multichain technology (Greenspan, 2015). Accordingly, each network node has a list with the addresses of all nodes. When a node disconnects, the remaining network nodes continue sending transactions to it until a limit of errors. In other words, when a node A receives a high number of denial of service from a node B, node A stops sending transactions to node B. On the other hand, when a node initialises, first, it triggers a message over the network informing its availability. Second, it tries to connect one of the other nodes from its configuration list. Figure 4 depicts the sequence diagram of this process. The new node joins the network after a handshake process with the already active nodes. In the initialisation process, a new node compares the timestamp of the last block stored locally with the timestamp of the previous block from the existing nodes. If necessary, it performs the consensus task, downloading the chunk of the history that it lacks locally. Only after this process can the new node listen to requests and perform operations in the blockchain network.

Figure 4 Sequence diagram of a scale-out operation



Notes: 1 ChainElastic manager turns an offline VM on
2 the offline VM starts the OS and the blockchain application
3 the blockchain application reaches one of the online VMs to join the network and download part of the database.

3.6 Definition of metrics

Although blockchain has specific metrics to assess its performance (Croman et al., 2016), we propose two additional metrics to evaluate the ChainElastic performance:

1 latency (L)

2 resource consumption per transaction (RpT).

Equation (3) demonstrates how the latency metric is computed. L defines the average of the latency of each transaction the system processes in a time interval. Let $transactions$ be the number of transactions completed between the instants t_0 and t_1, L is the result of the division of the time difference by the number of transactions.

$$L(t_0, t_1) = \frac{t_1 - t_0}{transactions} \tag{3}$$

Considering the resource consumption per transaction, RpT computes the average amount of resources needed to process a transaction in a time interval. Equation (4) demonstrates how this metric is computed. Let l be the total number of VM instances ($i + j$ according to Figure 2), and respectively t_0 and t_1 be the initial and final instants, then RpT calculates the amount of time the VM instances are in use in the interval $t_1 - t_0$. The final result is the division of this value by the number of transactions. $Time(m, t_0, t_1)$ returns the number of seconds the VM_m instance was active between the instants t_0 and t_1.

$$RpT(l, t_0, t_1) = \frac{\sum_{m=0}^{l-1} Time(m, t_0, t_1)}{transactions} \tag{4}$$

4 Evaluation methodology

In this section, we present all aspects related to the ChainElastic evaluation. In Subsection 4.1, the cloud infrastructure is described, and in Subsection 4.2 the prototype implementation is detailed. Finally, the evaluation parameters and scenarios are described in Subsection 4.3.

4.1 Infrastructure and prototype implementation

We deployed a private cloud environment using the OpenNebula middleware version 4.14.2. OpenNebula allows the user to set up a private cloud, offering flexibility to configure a controlled environment. Besides, the platform provides APIs for cloud administration, including scaling in and out operations. Our cloud is composed of five physical nodes with a 2.9 GHz dual-core processor, 4 GB of RAM, and an interconnection network of 100 Mbps. One of the nodes acts as the Cloud FrontEnd, whereas the remaining four act as physical cloud nodes in which VMs are deployed. All physical nodes have installed the Linux Ubuntu (https://ubuntu.com/) Server version 14.04.2 OS. We select the multichain system as the basis of our elasticity control. Multichain allows an easy to use interface, besides, to be user-friendly regarding configurations and parameters.

4.2 Prototype implementation

We developed the ChainElastic manager in Java using the OpenNebula Java API to connect to the cloud to manage resources. Through the API, the Manger performs all monitoring and elasticity operations. The prototype supports an XML SLA file containing its execution parameters. We created in the cloud a template for VM instances using a Linux Ubuntu Server version 17.04 OS. The template also defines fixed processing and memory resources for all instances. Therefore, the virtual infrastructure is composed of homogeneous VM instances for all elasticity operations. We also configured a VM disc image with a pre-installation of Multichain and attached it to the template.

Considering the user application, to produce computational load in the Multichain, we modeled a bash script that performs 1,250 sequential transactions to the Multichain, with a 32-bit hash size. These transactions consist of transferring values between different wallets. Each transaction triggers six mining operations, which result in the processing load.

4.3 Evaluation parameters and scenarios

The experiments comprise a combination of parameters with different executions scenarios. Considering that the cloud FrondEnd starts to acquire measures from resources at a defined interval, ChainElastic must use a monitoring interval higher than this. Although the OpenNebula default value is 60 seconds, we opted to change this parameter to five seconds to obtain updated values shortly. As the FrontEnd process of acquiring measures from resources may take some seconds according to the number of resources, we defined the ChainElastic monitoring interval to 15 seconds.

Regarding the elasticity parameters, we defined the upper and lower thresholds to 80% and 40%, respectively. Additionally, we determined the cool-down period in six observations. Therefore, at each elasticity operation, the user application has 90 seconds to adapt to the new resources configuration. Besides the parameters, we designed three different execution scenarios to compare results:

1 two fixed VM instances

2 eight fixed VM instances

3 starting with one VM and ChainElastic enabled.

Scenario 1 comprises the execution of the application with two online VM instances without enabling ChainElastic manager. Similarly, scenario 2 also does not have ChainElastic manager enabled; however, in this scenario, the number of online VM instances is eight. Finally, scenario 3 consists of the execution of the application with ChainElastic manager enabled, and starting with one online VM instance.

5 Results

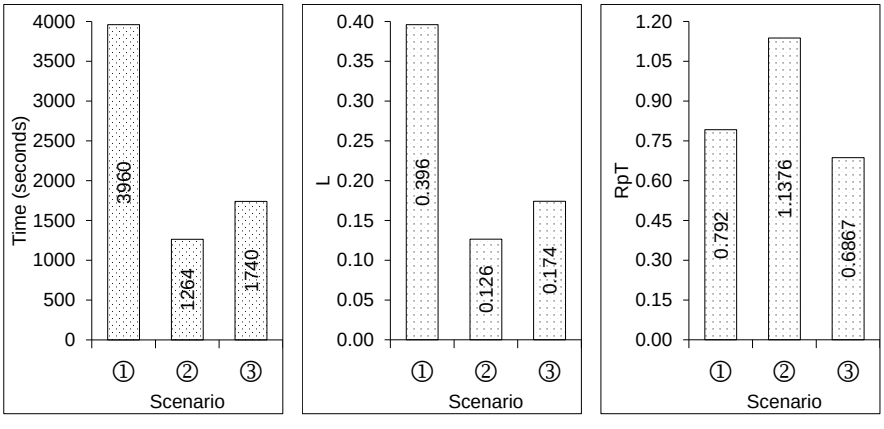
In this section, we describe the prototype evaluation into two sections. First, in Subsection 5.1 the performance evaluation according to the metrics defined in

Subsection 3.6 is presented. Then, in Subsection 5.2, the resource consumption profiles of each scenario are analysed.

5.1 Evaluating performance

Figure 5 illustrates the results of all metrics for all scenarios. Figure 5(a) shows the execution time relation between them. Scenario 2 achieved the best execution time due to a high number of available resources for workload distribution. In contrast, the low number of resources from scenario 1 produced the worst result. The results from scenario 3 demonstrate the influence of ChainElastic compared to scenarios with few (scenario 1) and many (scenario 2) resources. ChainElastic performed an execution time 37.66% higher than scenario 2, and, compared to scenario 1, it presented a decrease of 56.06%.

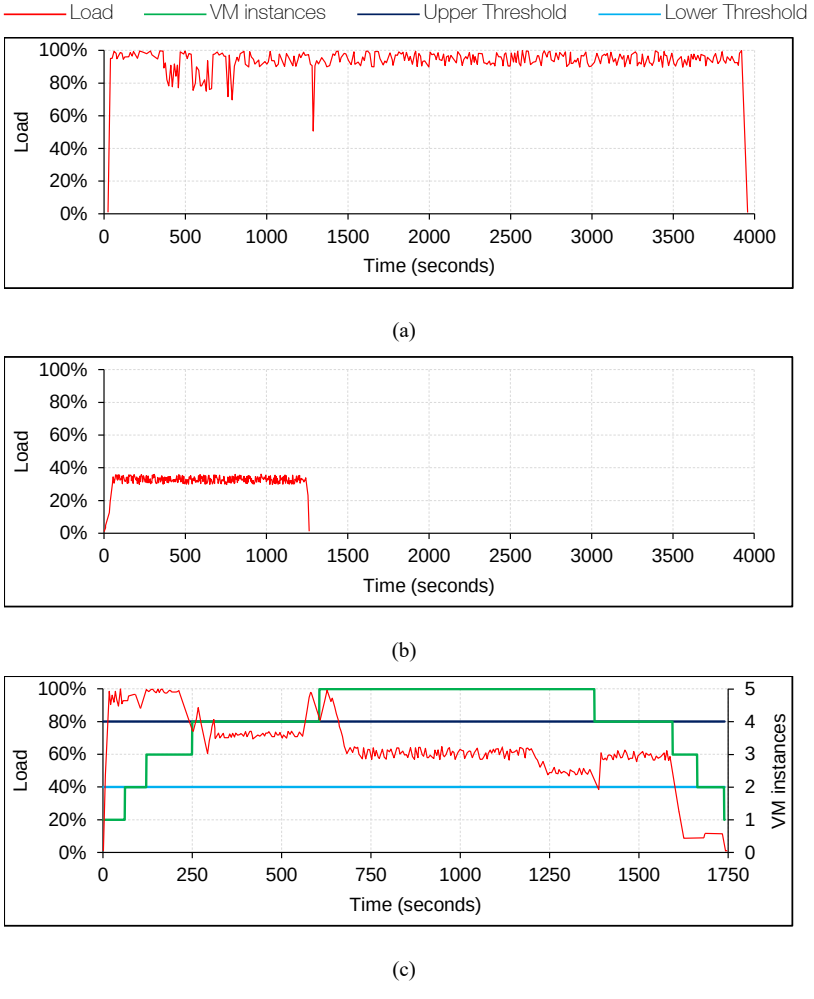
Figure 5 Results of, (a) metrics time ($Time$) (b) latency (L) (c) resource per transaction (RpT) from scenarios 1, 2, and 3



(a) (b) (c)

Figures 5(b) and 5(c) depict the results of metrics latency and resource per transaction for the three scenarios. Comparing 1 and 3, the automatic resource allocation strategies employed by ChainElastic resulted in decreases of 56.06% and 13.3% for the metrics L and RpT, respectively. Since ChainElastic achieved a better execution time than scenario 1, and because L is highly related to it, this result demonstrates the tremendous positive effect of ChainElastic. Additionally, even ChainElastic adding extra resources, scenario 3 achieved the best result of RpT. Comparing 3 and 4, the results are slightly different. On the one hand, ChainElastic obtained an execution time higher than scenario 2, which resulted in a 38.1% higher latency. On the other hand, the variable number of resources employed by ChainElastic achieved a decrease of 39.64% in the RpT metric compared to scenario 2.

Figure 6 System load profiles over the execution time from all scenarios, (a) scenario 1 (b) scenario 2 (c) scenario 3 (see online version for colours)



Notes: Figures 6(a) and 6(b) share the same time scale in the x axis.

5.2 Analysing resource profile

Figure 6 depicts the cloud resources load profile of all scenarios. Both Figures 6(a) and 6(b) have the same time and load scale to simplify the comparison of fixed resource scenarios. On the one hand, scenario 1 took 3,960 seconds to compute all transactions with only two VM instances available. Most of the time, the load is above 90%, showing the available resources in an overload state when processing the transactions. On the other hand, scenario 2 achieved an execution time of 1,264 seconds (68% better) when processing all transactions with four times more VM instances. A higher number of resources results in better load distribution, which directly affects the execution time. However, with a shorter workload for each VM instance, the system's average load remained below 40%, causing a waste of resources.

Figure 6(c) illustrates the load and resource profiles of scenario 3. Differently from the other two, here, the number of resources varies with time. ChainElastic performs scale in and out operations according to the load level compared to the thresholds. Within the first 250 seconds, the load is above the upper threshold, which triggers scale-out operations until the system load decrease below this threshold. As a result of these operations, the number of resources increases from one to four VM instances. Consequently, more resources result in better load distribution, leading to a decrease between 70% and 75%. At the end of the execution, due to the decreasing load, ChainElastic triggers four scale-in operations bringing the number of available resources to the original state.

6 Conclusions

Private blockchain provides reliability on stored data in the IoT scope. Its use in this scope is essential to offer warranties for stakeholders of industries and companies. However, up to now, we suffer from both:

1 manual management of resource to support IoT-based blockchain applications

2 the need completely to boot from scratch a new computing node on scaling up operations.

In this context, this study presented ChainElastic, a model for automatic resource provisioning in the cloud for IoT-based private blockchain applications. We enabled our contributions through a framework that makes available:

1 a reactive-based elasticity manager

2 the use of active and inactive nodes to support partial data updating when enlarging the infrastructure resources.

In step 1, decision-making consists of evaluating the current load of all VMs, thus comparing it against lower and upper thresholds to launch or not resource reorganisation.

Experiments demonstrate that ChainElastic improves application performance and system resource consumption. The results show gains up to 56.06% in the execution time when comparing elastic and non-elastic scenarios. Also, 30.64% were the gains in resource consumption in favour of elastic deployments. As said, encouraging results were obtained with the current prototype that used the CPU metric in the elasticity manager calculus. Future work envisions the use of several metrics in this scope, combining IO – network, memory, and disk, for example and CPU-bound metrics for better accuracy on resource reorganisation operations.

Acknowledgements

This work was partially supported by the following Brazilian Agencies: FAPERGS, CAPES, and CNPq (303640/2017-0).

References

Al-Haidari, F., Sqalli, M. and Salah, K. (2013) 'Impact of CPU utilization thresholds and scaling size on autoscaling cloud resources', in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, December, Vol. 2, pp.256–261.

Bodkhe, U., Tanwar, S., Parekh, K., Khanpara, P., Tyagi, S., Kumar, N. and Alazab, M. (2020) 'Blockchain for Industry 4.0: a comprehensive review', *IEEE Access*, Vol. 8, pp.79764–79800.

Bruce, J.D. (2014) *The Mini-Blockchain Scheme* White Paper.

Chen, I., Guo, J., Wang, D., Tsai, J.J.P., Al-Hamadi, H. and You, I. (2019) 'Trust-based service management for mobile cloud IoT systems', *IEEE Transactions on Network and Service Management*, March, Vol. 16, No. 1, pp.246–263.

Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E.G., Song, D. and Wattenhofer, R. (2016) 'On scaling decentralized blockchains', in Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M. and Rohloff, K. (Eds.): *Financial Cryptography and Data Security*, pp.106–125, Springer, Berlin, Heidelberg.

Dorri, A., Kanhere, S.S., Jurdak, R. and Gauravaram, P. (2019) 'LSB: a lightweight scalable blockchain for IoT security and anonymity', *Journal of Parallel and Distributed Computing*, Vol. 134, pp.180–197.

Eyal, I., Gencer, A.E., Sirer, E.G. and Van Renesse, R. (2016) 'Bitcoing: a scalable blockchain protocol', in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation, NSDI'16*, USENIX Association, Berkeley, CA, USA, pp.45–59.

Furtado, F.R., Silva e Silva, J.V., Cappellari Jr., M., Castilhos, C.H.M., Rodrigues, V.F., Da Costa, C.A. and Da Rosa Righi, R. (2020) 'Towards characterising architecture and performance in blockchain: a survey', *International Journal of Blockchains and Cryptocurrencies*, Vol. 1, No. 2, pp.121–153.

Galante, G. and de Bona, L.C.E. (2012) 'A survey on cloud computing elasticity', in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, UCC'12*, IEEE Computer Society, Washington, DC, USA, pp.263–270.

Gao, F., Zhu, L., Shen, M., Sharif, K., Wan, Z. and Ren, K. (2018) 'A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks', *IEEE Network*, November, Vol. 32, No. 6 pp.184–192.

Greenspan, G. (2015) *Multichain Private Blockchain*, White Paper [online] http://www.multichain.com/download/MultiChain-White-Paper.pdf.

Herbst, N.R., Huber, N., Kounev, S. and Amrehn, E. (2013) 'Self-adaptive workload classification and forecasting for proactive resource provisioning', in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, ICPE'13*, ACM, New York, NY, USA, pp.187–198.

Johng, H., Kim, D., Park, G., Hong, J-E., Hill, T. and Chung, L. (2020) 'Enhancing business processes with trustworthiness using blockchain: a goal-oriented approach', in *Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC'20*, Association for Computing Machinery, New York, NY, USA, pp.61–68.

Killer, C., Rodrigues, B., Matile, R., Scheid, E. and Stiller, B. (2020) 'Design and implementation of cast-as-intended verifiability for a blockchain-based voting system', in *Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC'20*, Association for Computing Machinery, New York, NY, USA, pp.286–293.

Kuhlenkamp, J., Werner, S., Borges, M.C., Ernst, D. and Wenzel, D. (2020) 'Benchmarking elasticity of FaaS platforms as a foundation for objective-driven design of serverless applications', in *Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC'20*, Association for Computing Machinery, New York, NY, USA, pp.1576–1585.

Lee, Y., Avizienis, R., Bishara, A., Xia, R., Lockhart, D., Batten, C. and Asanović, K. (2011) 'Exploring the tradeoffs between programmability and efficiency in data-parallel accelerators', in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pp.129–140.

Lee, X.T., Khan, A., Gupta, S.S., Ong, Y.H. and Liu, X. (2020) 'Measurements, analyses, and insights on the entire ethereum blockchain network', in *Proceedings of the Web Conference 2020, WWW'20*, Association for Computing Machinery, New York, NY, USA, pp.155–166.

Liu, H., Zhang, Y. and Yang, T. (2018) 'Blockchain-enabled security in electric vehicles cloud and edge computing', *IEEE Network*, May, Vol. 32, No. 3, pp.78–83.

Mayergoyz, I.D. (2003) *Mathematical Models of Hysteresis and their Applications*, Academic Press.

Mireslami, S., Rakai, L., Far, B.H. and Wang, M. (2017) 'Simultaneous cost and QoS optimization for cloud resource allocation', *IEEE Transactions on Network and Service Management*, September, Vol. 14, No. 3, pp.676–689.

Moin, S., Karim, A., Safdar, Z., Safdar, K., Ahmed, E. and Imran, M. (2019) 'Securing IoTs in distributed blockchain: analysis, requirements and open issues', *Future Generation Computer Systems*, Vol. 100, pp.325–343.

Morktexs, V.J., Paschen, J. and Boon, E. (2019) 'How blockchain technologies impact your business model', *Business Horizons*.

Ozisik, A.P., Andresen, G., Bissias, G., Houmansadr, A. and Levine, B. (2017) 'Graphene: a new protocol for block propagation using set reconciliation', in Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H. and Herrera-Joancomartí, J. (Eds.): *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pp.420–428, Springer International Publishing, Cham.

Reyna, A., Martín, C., Chen, J., Soler, E. and Díaz, M. (2018) 'On blockchain and its integration with IoT challenges and opportunities', *Future Generation Computer Systems*, Vol. 88, pp.173–190.

Righi, R.d.R., Rodrigues, V.F., da Costa, C.A., Galante, G., de Bona, L.C.E. and Ferreto, T. (2016) 'Autoelastic: automatic resource elasticity for high performance applications in the cloud', *IEEE Transactions on Cloud Computing*, January, Vol. 4, No. 1, pp.6–19.

Rizun, P. (2016) 'Subchains: a technique to scale bitcoin and improve the user experience', *Ledger*, Vol. 1, pp.38–52.

Seigneur, J-M., Pusterla, S. and Socquet-Clerc, X. (2020) 'Blockchain real estate relational value survey', in *Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC'20*, Association for Computing Machinery, New York, NY, USA, pp.279–285.

Sompolinsky, Y. and Zohar, A. (2015) 'Secure high-rate transaction processing in bitcoin', in Böhme, R. and Okamoto, T. (Eds.): *Financial Cryptography and Data Security*, pp.507–527, Springer, Berlin, Heidelberg.

Wang, E.K., Liang, Z., Chen, C-M., Kumari, S. and Khan, M.K. (2020) 'PORX: a reputation incentive scheme for blockchain consensus of IoT', *Future Generation Computer Systems*, Vol. 102, pp.140–151.

Wu, L., Garg, S.K. and Buyya, R. (2015) 'Service level agreement (SLA) based saas cloud management system', in *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, December, pp.440–447.

Zhang, Q., Cheng, L. and Boutaba, R. (2010) 'Cloud computing: state-of-the-art and research challenges', *Journal of Internet Services and Applications*, May, Vol. 1, No. 1, pp.7–18.

Zhong, Z. and Buyya, R. (2020) 'A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources', *ACM Trans. Internet Technol.*, April, Vol. 20, No. 2.