applied sciences

MDPI

Article

Lightweight Blockchain Processing. Case Study: Scanned Document Tracking on Tezos Blockchain

Mohamed Allouche [1], Tarek Frikha [2,*], Mihai Mitrea [1], Gérard Memmi [3] and Faten Chaabane [4]

[1] ARTEMIS Department, Telecom SudParis-Institut Polytechnique de Paris, 91011 Palaiseau, France; mohamed.allouche@telecom-sudparis.eu (M.A.); mihai.mitrea@telecom-sudparis.eu (M.M.)
[2] CES Lab, Université de Sfax, Sfax 3038, Tunisia
[3] Telecom Paris-Institut Polytechnique de Paris, LTCI, 91011 Palaiseau, France; gerard.memmi@telecom-paris.fr
[4] Regim Lab, Université de Sfax, Sfax 3038, Tunisia; chaabane.faten@gmail.com
* Correspondence: tarek.frikha@enis.tn; Tel.: +216-582-9911

Featured Application: This paper establishes the proof-of-concepts for using on-chain/off-chain load balancing as an enabler for Blockchain deployment on lightweight computing resources. In this way, a Block-chain of an arbitrarily large number of nodes can be deployed over any combination of com-puting resources, from cloud servers and PCs to Raspberry Pi 3. While the illustrations corre-spond to visual content tracking (fingerprinting), the advanced on-chain/off-chain load bal-ancing approach can potentially serve any other multimedia application (of comparable com-plexity) that requires synergies with Blockchain solutions.

Abstract: To bridge the current gap between the Blockchain expectancies and their intensive computation constraints, the present paper advances a lightweight processing solution, based on a load-balancing architecture, compatible with the lightweight/embedding processing paradigms. In this way, the execution of complex operations is securely delegated to an off-chain general-purpose computing machine while the intimate Blockchain operations are kept on-chain. The illustrations correspond to an on-chain Tezos configuration and to a multiprocessor ARM embedded platform (integrated into a Raspberry Pi). The performances are assessed in terms of security, execution time, and CPU consumption when achieving a visual document fingerprint task. It is thus demonstrated that the advanced solution makes it possible for a computing intensive application to be deployed under severely constrained computation and memory resources, as set by a Raspberry Pi 3. The experimental results show that up to nine Tezos nodes can be deployed on a single Raspberry Pi 3 and that the limitation is not derived from the memory but from the computation resources. The execution time with a limited number of fingerprints is 40% higher than using a classical PC solution (value computed with 95% relative error lower than 5%).

Keywords: Blockchain; Tezos; on-chain/off-chain load balancing; multiprocessor architecture

Citation: Allouche, M.; Frikha, T.; Mitrea, M.; Memmi, G.; Chaabane, F. Lightweight Blockchain Processing. Case Study: Scanned Document Tracking on Tezos Blockchain. *Appl. Sci.* **2021**, *11*, 7169. https://doi.org/10.3390/app11157169

Academic Editor: Gianluca Lax

Received: 31 May 2021
Accepted: 28 July 2021
Published: 3 August 2021

1. Introduction

In the context of the 2007–2008 worldwide financial crisis, Blockchains emerged as an attempt to ensure trust and to enhance the security side of financial transactions in peer-to-peer networks. Blockchains are specialized computing machines, optimized to perform a reduced set of repetitive operations related to user/message/transaction authentication, and achieved by public-key protocols, hash computation, and consensus protocols. Their potential benefits, including decentralized trust and resilience, are generally put into balance against their highly intensive computing requirements. Illustrating this point, it was reported that in 2008 the BitCoin [1] energy consumption was 16% larger than the whole energy consumption of Ireland [2].

Since then, Blockchain is no longer intrinsically connected to financial applications and has gradually become an appealing solution for a large variety of business verticals, such

as industry 4.0 [3], logistics [4], agriculture [5], e-health [6], and fake news detection [7], to mention but a few. Yet, the applicative needs of such verticals are generally catered for by conventional, general-purpose computing machines, increasingly accommodated by elastic cloud resources.

Faced with these challenges, conventional Blockchain architectures reach their inner limitations and often become prohibitive in terms of memory, computation and energy-resources consumption required to achieve a predefined task into a limited time lapse [8]. Moreover, a second type of limitation relates to the complexity of code development: each Blockchain comes across with specific programming languages and with different syntactic constraints.

To jointly solve these limitations and having in view a lightweight generation of Blockchains, the present paper advances a distributed on-chain/off-chain architecture, allowing the global applicative logical workload to be securely distributed between on-chain and off-chain resources, while keeping the global orchestration of the underlying execution processes. We understand off-chain resources to mean: a general-purpose computing machine with conventional computing and storage resources (elastic or not) and with a generic programming language (e.g., C or Python).

The applicative illustrations correspond to the Tezos Blockchain and to a scanned document tracking system based on visual fingerprinting. Note that scanned documents are characterized by their inner semantic/digital representation duality: actually, a unique document, from the point of view of human understanding, can have multiple digital representations, as illustrated in Figure 1. This is the result of the fact that the scanning conditions are never identical and, even in would-be identical conditions, the physical characteristics of each sensor result in a unique scanned document. Consequently, digital documents cannot be directly tracked by Blockchains (for which a single bit change in the digital representation corresponds to a new document) and solutions related to visual fingerprinting should also be investigated. Visual fingerprints are compact and salient visual content features computed from the content itself, which can uniquely identify duplicated and/or replicated versions of it in a reference database. It is also referred to as content-based copy detection (CBCD) or near duplicate detection [9].

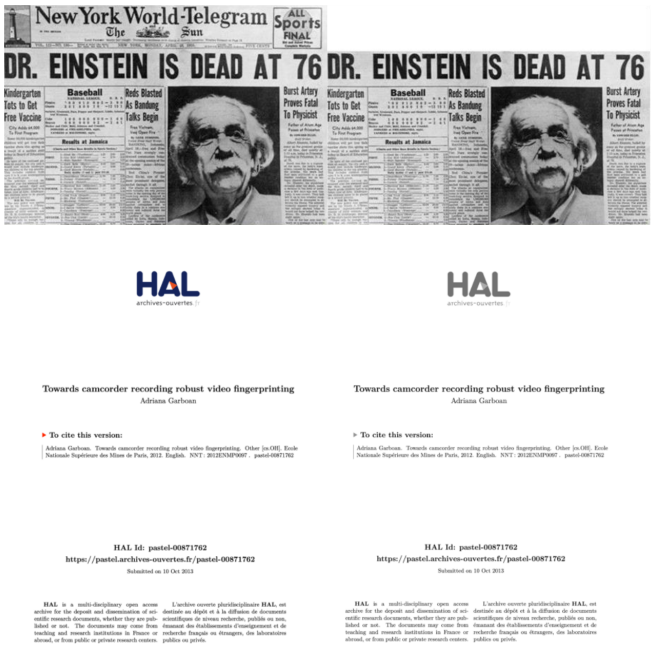

Figure 1. Digital documents: heterogeneous content (text, image, etc.) with various digital representations for the same semantic content.

In a nutshell, the main contributions of the paper relate to the methodological, experimental and applicative levels, as follows:

- methodological level: conception, specification and implementation of an on-chain/off-chain load-balancing solution, thus making it possible for the intimately constrained computing, storage and software resources of any Blockchain to be abstractly extended by general-purpose computing machine resources;
- experimental level: specifying an experimental testbed and carrying out the underlying experiments for achieving the proof-of-concepts for complex Blockchain application execution (namely visual fingerprinting) on lightweight computing resources (namely, a multiprocessor ARM embedded platform, integrated into a Raspberry Pi);
- applicative level: the methodological framework developed in this study makes it possible for a Blockchain of an arbitrarily large number of nodes to be deployed over any combination of computing resources, from cloud servers and PCs to Raspberry Pi; in this way, even low resource devices (Raspberry Pi) can host up to nine nodes executing complex applications (namely visual fingerprinting).

The paper is structured as follows: Section 2 analyses the state-of-the-art Blockchain applicative perimeter and introduces some basic notions related to visual fingerprinting and to lightweight/embedded architectures. Section 3 is devoted to the presentation of the advanced solution. Section 4 describes the applicative set-up, while Section 5 presents the experimental results. Section 6 discusses the research results. Section 7 concludes the paper and identifies the short-term perspectives for our future work.

2. State-of-the-Art

2.1. Blockchain at a Glance

Be Alice and Bob, two out of the N users in a network, exchanging peer-to-peer messages in an authenticated and decentralized way. The messages sent by Alice to Bob are first authenticated by one or a group of users in the network, according to a protocol everybody agreed on. Once authenticated, the message is stored (together with the Alice's and the authenticator's IDs) into a ledger that is distributed to all the users. Assuming a malicious user or a group of users wants to modify a local copy of the ledger, the other users can identify this modification and are able to fix it by recovering the unaltered version of the ledger.

The paragraph above is the Blockchain narrative. Note that in this description, the actual nature of the messages is not relevant; they can be related to digital currency transfer (as initially thought) or to any other kind of information (permission granting process, notification, etc.). According to [10,11], the Blockchain is an information storage and distribution technology, which is transparent, secure, and operates without a central control body. By extension, a Blockchain is a database that contains the history of all the exchanges made among its users, since its creation. This database is secure and distributed; it is shared by its different users, without intermediaries, thus allowing any authorized user to check the validity of the information chain.

Transactions between users are grouped in blocks. Each block is validated by the nodes of the network (e.g., the so-called miners or bakers). Once validated, the block is timestamped and added to the Blockchain. Then the transaction becomes visible not only to the receiver, but also to the entire network.

Blockchains consider the so-called consensus algorithms to preserve their security side. In this context, several types of consensus algorithms are proposed in the literature such as the Proof of Work (PoW), the Proof of Stake (PoS), or the Proof of Authority (PoA); they are considered by themselves or in conjunction with some optimization mechanisms, for instance, the Practical Byzantine Fault Tolerance (PBFT). One of the most complex and energy-intensive consensus solutions is PoW that was used in several Blockchains, such as Bitcoin, Ethereum and IoTA [12]. To reduce computation requirements while keeping a prescribed level of security, the PoS solutions have been advanced and are currently

considered by Blockchains, such as Tezos, Peercoin, Nxt, Blackcoin, or ShadowCoin, to mention but a few.

A Smart contract is a software installed on the Blockchain that automatically executes a logical work description that is a pre-programmed contractual commitment. It is not a legal document in itself, but it automates the execution of a contractual commitment. Each Blockchain comes with its Smart contract programming language, as for example the Solidity language for Ethereum, the Go language for Hyperledger, or the Michelson language for Tezos. Both similarities and differences among such programming languages exist; the differences might relate to the data types, to the instruction set, to the syntax, to the concept of transaction fee, etc.

The intensive computing behavior of the Blockchains is derived from both the consensus algorithm and from the execution of the complex operations, (at least) logically imbricated with Smart contracts execution.

For instance, the PoW algorithm is known to be a priori incompatible with lightweight implementation. Thus, it was shown in [13,14] that an Ethereum-based system cannot be implemented on a Raspberry Pi 3 platform. The implementation of Ethereum PoW on the ARM (Advanced RISC Machine) architecture of the Raspberry Pi 3 made the platform crash, even when a single node is implemented. In contrast, such an implementation is possible when using a Field-Programmable Gate Array (FPGA) platform to replace GPUs for consensus implementation. While such state-of-the-art limitation relates to existing Blockchain solutions, early PoS prototype algorithms with nodes deployed on a Rasberry Pi have also been recently advanced [15]. Yet, to the best of our knowledge, no study has been reported on the deployment of PoS algorithms on Raspberry Pi.

In order to make provision for load-balancing in Smart contract execution, the reference architecture of Enterprise Ethereum Alliance (EEA) considers both on-chain and off-chain processing and storage [16]. According to the EEA principles, the load-balancing is expected to dynamically occur, according to the actual level of processing.

In the present paper, we will advance an on-chain/off-chain load balancing solution for PoS Blockchains and we will demonstrate that in this way, a complex multimedia operation (scanned document fingerprinting) can be deployed on a Raspberry Pi 3 platform. Note, we will not modify the existing consensus algorithms, thereby, keeping the initial Blockchain wise security level.

2.2. Video Fingerprinting Basic Concepts

Video fingerprinting methods are meant to identify duplicated and slightly replicated versions of a given video sequence (query) in a reference video database [17–19]. They are also commonly referred to as content-based copy detection (CBCD) or near duplicated content detection.

Any fingerprinting method encompasses two main steps. Firstly, some visual content-based features (fingerprints) that are able to concisely and accurately represent that query are computed. Secondly, such features should be matched with the reference database, according to a similarity metric and a threshold pre-set using functional and/or theoretical criteria.

Fingerprinting systems are characterized by three main properties. Firstly, the uniqueness property requires for two video sequences with different content to have two different fingerprints (in the sense of the similarity measure and of its underlying threshold). Secondly, the robustness property requires that any query video sequence and its replicas obtained through different distortions have identical fingerprints (here again, in the sense of the similarity measure and of its underlying threshold). Finally, the scalability property relates to the system's ability to deal with large databases.

From the methodological point of view, fingerprinting computation can be achieved by various means [9]; we mention here-after only a few: 2D-DWT (Discrete Wavelet Transform) coefficients, 3D-DCT (Discrete Cosine Transform) coefficients, pixel differences between consecutive frames, visual attention regions, quantized block motion vectors,

MPEG-7 features, such as the color-layout descriptor, invariant moments of frames edge representation, dominant-edge orientation of keyframes. For each and any fingerprint, a different matching rule is devised so as to ensure the check of any potential suspect video sequence against a (massive) prerecorded database.

Hence, the complexity of visual fingerprinting applications is given both by the fingerprinting computation and by its matching. In the present paper, the focus is set on the way in which such a visual fingerprinting application can be coupled to Blockchain, while ensuring an end-to-end lightweight implementation, demonstrated by a deployment on an embedded platform. Consequently, we will not design a new fingerprinting method, but will integrate an existing one, based on DWT coefficients [9] and designed for tracking camcorder video sequences.

2.3. Multimedia Multiprocessor Embedded Architecture

The multimedia application deployment on embedded systems has undergone a remarkable evolution, and different platforms have become adequate and efficient for the implementation of different applications. While some platforms have been considered for implementation and testing, others have enabled prototyping and the implementation of optimized architectures for multimedia applications.

The existence of parallel applications has increased the importance of multi-core processors that resulted in the emergence of GPGPUs (General Purpose GPUs). The Graphics Processing Unit is a coprocessor, originally developed to accelerate graphics applications, and that consists of hundreds of cores and is capable of processing thousands of threads. The cores of a GPU execute the same instruction sequence but possibly on different data elements, thus turning the GPU into a predilection solution for a large variety of applications, particularly involving matrices, such as deep learning, machine learning, augmented reality, or virtual reality, to mention but a few. Yet, GPUs intrinsically come across with high energy consumption [20], thereby becoming unsuitable for a lightweight solution we are targeting in the present paper.

Consequently, alternative solutions are searched for, and the FPGA family is a good candidate (note that server manufactures such as Intel have also preferred FPGAs for their lower energy consumption [21]).

In fact, with the advent of heterogeneous computing systems such as the Xilinx Zynq UltraScale+ multiprocessor system-on-chip (MPSoC) [22], different processing units can be embedded in the System-on-Chip (SoC) to meet the growing requirements of the applications (performance/energy constraints). Moreover, hardware solutions offered by FPGA platforms provide the expected speed-up, while reducing the energy consumption. By connecting an FPGA-based accelerator to a programmable Central Processing Unit (CPU), performance and energy benefits are obtained while retaining more flexibility than an Application Specific Integrated Circuit (ASIC).

For instance, in [23], the embedded experimentation of the Viola Jones face detection algorithm is based on a mixed HW/SW architecture; in this way, thanks to the processing parallelism, a system more stable than a classical PC is obtained. For an identical stability, the implementation of a massively parallel architecture on FPGA becomes faster.

Compared to GPU, FPGA is certainly less efficient, but its energy consumption is lower. The use of a heterogeneous architecture based on parallel processors with a hardware IP was presented in [24]. The application is implemented on a multi-core Zynq platform using dual ARM processors. The face detection application provided a 7.8 times gain, compared to a classical dual ARM processor architecture.

Raspberry Pi is a particular case of a multiprocessor ARM embedded platform that, despite its low computation and storage resources, continuously gains popularity thanks to its low cost and its use in education/academic programs. In [25], a new face detector adapted to on-board systems has been proposed. Based on the construction of face detection networks with low computational cost and sufficient capacity, this efficient method uses convolution factorization to build the network with scattered connections. The EagleEye

runs on the embedded device based on the ARM Cortex-A53 (Raspberry Pi1 3b+ [26–29]) at 21 FPS with the input of VGA resolution with better functional accuracy than methods with the same order of computational complexity.

This concise state-of-the-art brings to light that FPGA is considered today's mainstream embedded solutions for multimedia applications, and that Raspberry Pi is the challenger. Yet, the applicative borders between FPGA and Raspberry Pi are not clearly drawn. Our paper studies whether a lightweight implementation of a PoS Blockchain supporting a video fingerprinting application can be deployed on a Raspberry Pi 3 or not.

3. The Advanced Solution: On-Chain/Off-Chain Load Balancing

This section presents the on-chain/off-chain load balancing solution that we advanced in order to reduce the computation effort on the Blockchain side. As the overall computation task assigned to the Blockchain is unitary from the logical point of view, we will also keep a unitary logical execution while balancing on-chain/off-chain the operations. Hence, the advanced solution should be tightly coupled to the Smart contract execution flow.

In the sequel, we will present this solution for the particular case of the Tezos Blockchain [30,31]: however, this is undertaken for the sake of clarity and the concept is independent with respect to the Blockchain peculiarities.

3.1. Conventional Smart Contract Workflow

Tezos is a Smart contract platform, featuring formal verification for Smart contracts: a mathematical proof that all deployed Smart contracts are designed and executed according to their design is thus provided.

While the formal verification is a precious tool to help Smart contract programmers, the underlying Smart contract deployment is still a complex process that requires several steps, as illustrated in Figure 2 and explained here-after:

1. The Programming step writes a Michelson program that converts a logical contract (if . . . then series) into a Tezos script.
2. The Formal Verification step executes the Tezos command "typecheck script" (with the proper parameters); this command does not apply any action on the Blockchain, yet its output informs the developer about the Smart contract correctness.
3. The Deployment step executes the Tezos command "originate" with the proper parameters (Smart contract name, owner, transaction cost, ...). The Smart contract is added to the new block (the Smart contract keys are generated) and is ready to be invoked.
4. The Call/Execute step executes the Tezos command "transfer" with its parameters (Smart contract ID, Smart contract monetizing conditions, Smart contract parameters, . . .); the action is added to the Blockchain, the Smart contract is executed, and the results are stored in the Blockchain.

While this basic Smart contract deployment workflow perfectly caters to the needs of legacy, financial-related operations (a simple transfer from one account to another, assuming a given condition is met), their limitations appear when dealing with more complex-use cases.

For instance, the Tezos' Michelson language is bound to process integer values and cannot be extended (while keeping backward compatibility) so as to process floating point values.

Moreover, as an example of applicative limitation, the Blockchain operation cost steadily increases with the size and complexity of the Smart contract. In order to overcome these limitations, two solutions can be considered.

First, the mapping of the operations from floating point to integers can be a solution for this fundamental limitation but it increases both the size and the computational complexity of the Smart contracts.

Consequently, we designed an architecture for ensuring an on-chain/off-chain code balancing, as described in the next section.

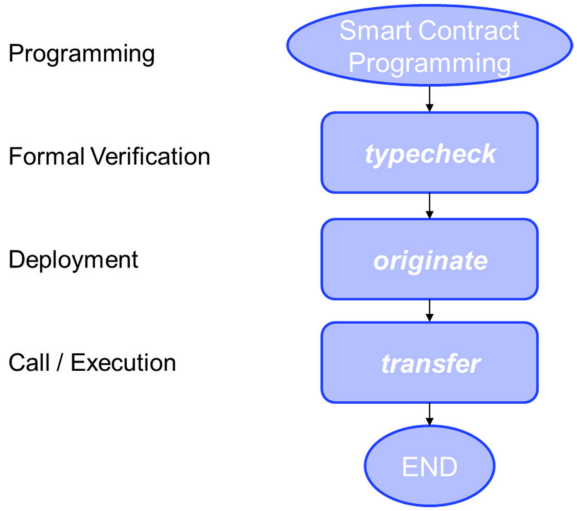

Figure 2. Tezos Smart Contract Workflow.

3.2. New Smart Contract Workflow of On-Chain/Off-Chain Balancing

In order to overcome the conventional Smart contract limitations (see Figure 2) for complex applications, we designed the architecture presented in Figure 3.

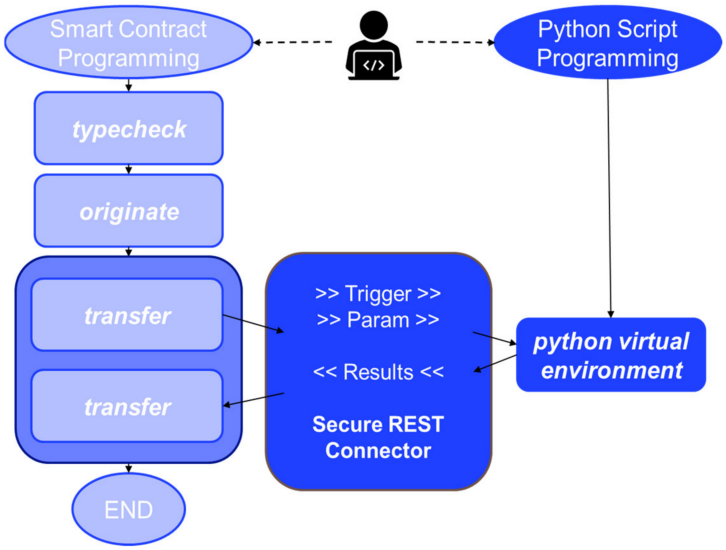

Figure 3. On-chain/off-chain code balancing.

The new architecture ensures that the computational load, which is formally described as a list of tasks to be fulfilled by the Blockchain-based application, is balanced between on-chain and off-chain computing.

On the one hand, the Blockchain Smart contract code is generated to take charge of the transactional aspects (monetizing, legal articles) of the global task. It is also in charge of the orchestration (synchronization) of the actions.

On the other hand, a general-purpose computing machine (a Python machine, in our example) is deployed for executing the complex operations that are not Blockchain natives. The corresponding code (written in Python, in our example) is paired with the transactional Smart contract, and it is executed under its control.

The connector serves as an entry and exit point for both on-chain and off-chain entities. Firstly, it is a tool that scans the Blockchain state, block generation and transaction made in order to notify the off-chain entity whenever an action or intervention is requested. Secondly, it collects the required data (block ID, transaction ID, needed action, parameters,

. . .) and passes them to the external (Python) script. From the external script point of view, this data is the invoker of the entry parameters and the execution. The script decodes the entry parameters, parses them, executes the requested operations and encodes the results. The encoded result is sent back to the connector that will first check the authenticity and integrity and then transfer it to the on-chain entity. Finally, the resulting data is added to the Blockchain as a proof of accomplishment. The connector transfers the results to the Smart contract that finishes its execution. The connector has access to both on-chain and off-chain entities thanks to a REST API (the Tezos node provides a JSON/RPC interface [32] and we developed the off-chain part).

Of course, in such a system that mixes multiple processing environments, the data communication between the entities is usually considered as a security failure point, and particular attention should be paid in this respect. Firstly, the data exchanged through the connector are encrypted by an https mechanism: in this way, the off-chain virtual machine is controlled only by trusted entities. Secondly, specific security constraints are set to Smart contract programming, in order to ensure the integrity and the non-repudiation of all accepted requests; implicitly, an attacker that was able to pass the https level of security is thereby referred from rewriting the chain. Thus, the contract will record whatever actions trigger its entry points. Moreover, if an outcome in the contract depends on at least one user argument, the contract will not accept any additional user-supplied inputs until the off-chain process finishes the in-progress tasks.

4. Experimental Set-Up

This section describes the experimental set-up; it is structured into three sub-sections, accordingly, the hardware/software background for experiments, the presentation of the on-chain/off-chain solution code, and the fingerprinting method and database, respectively.

4.1. Hardware/Software Experimental Platform

To validate the efficiency of the proposed solution, a multiplatform deployment was considered to jointly offer the user the simplicity of graphic user interfaces and the size and low-energy consumption of an embedded platform.

The hardware part consists of a PC (acting as an interface) and a Raspberry Pi (acting as the computing device). The PC has the following characteristics: CPU of 64-bit 6-core Intel Xeon E5-1650, 3500 MHz, GPU of (2x) AMD FirePro W2100-2GB DDR3, RAM: 32 GB, 2133MHz, with dual Ubuntu/Windows OS. The Raspberry Pi has the following configuration: CPU of 64-bit quad-core ARM Cortex-A53, 1200 MHz, GPU: Multimedia VideoCore IV, 400 MHz, RAM 1 GB SDRAM LPDDR2, 900 MHz, operated by Raspbian Buster Lite (ver. Sept. 2019), kernel version 4.19.

The task allocation on these two computing devices is illustrated in Figure 4. The PC acts just as a kind of interface and it is in charge with the ingestion of the visual documents. The rest of the computing tasks are performed on the Raspberry Pi (both fingerprinting processing and on-chain/off-chain processing).

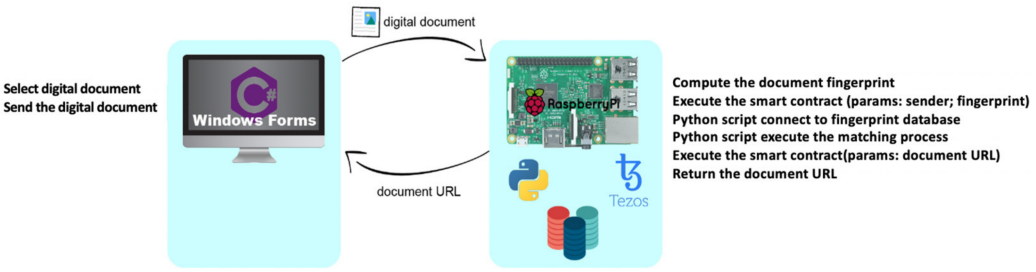

Figure 4. Task allocations on the two computing devices.

On the Raspberry Pi, the document fingerprint is first extracted by a devoted software. Then, the on-chain/off-chain load balancing module is invoked. This module starts by

an on-chain task, namely performing the first part of the Smart contract (parameters such as sender and fingerprint). Then, the off-chain part is invoked, and the Python script checks the fingerprint's existence in the database and, if not present, adds it to the database. Finally, the on-chain part executes the last part of the Smart contract.

4.2. On-Chain/Off-Chain Load Balancing Code

The software implementation is illustrated by the execution sequence (cf. Figure 5) as well as by some excerpts from the code executed on the Raspberry Pi. The complete software implementation is made available as open source under Apache license.

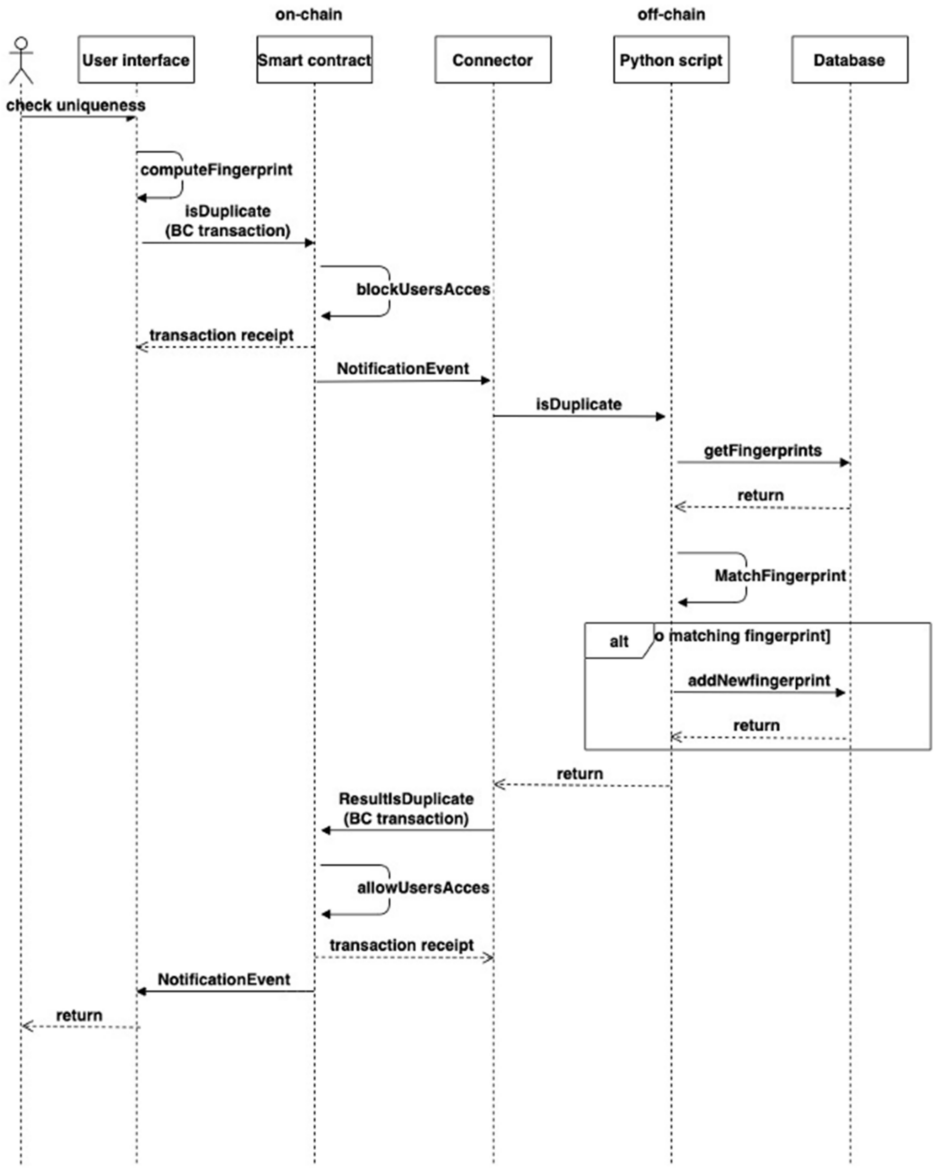

Figure 5. Execution sequence diagram.

We assume that the query fingerprint is already extracted (off-chain) and the user verifies whether it is present in the blockchain or not.

While the sequence diagram in Figure 5 shows the life-cycle of the fingerprinting application, it can be considered as a generic workflow for our on-chain/off-chain load balancing.

When the user wants to check the uniqueness of a digital document, the fingerprint computation is launched through the graphic interface. Then, a transaction to the Smart contract address precising the entry point and passing the fingerprint as an argument is

made. As soon as the Smart contract is triggered, it blocks the access of other users to avoid conflicts and avoid any potential security issues. When the block containing that transaction is processed and validated (e.g., by the miner or baker), the user receives the transaction receipt containing its details and the connector receives a notification event that is subscribed to. The connector fetches the user's request and forwards it to the Python script where the document fingerprint is compared with all previous fingerprints processed by the system, and then returns the matching process results to the connector. At this stage, the connector uses a moderator account to make a transaction to the Smart contract with the results. The Smart contract authorizes the moderator transaction, allowing the user to access again and notify the user via an Event.

This process is illustrated here-after through 4 code samples, labeled by (a)–(d).

(a) On-chain: user initializes a Tezos node and transfers the operation fees from its address to the Smart contract address (with the required arguments)

```
# User:  connect to tezos blockchain
$ ./tezos-init-sandbox-client.sh
#User:  call the 'left' smart contract entry point by making a
#transaction to the smart contract address with the required fees
$ tezos-client transfer $PRICE from $CLIENT_ADDRESS to $SMART_CON
TRACT_ADDRESS -arg "Left (Pair "$DOCUMENT_NAME" "$DOCUMENT_FINGERPR
INT")" -burn-cap 20
```

(b) On-chain: user inquires the Blockchain status

```
#> if the transaction passes successfully:
# > the smart contract lock further access to that entry point
# > a blockchain scanner notifies the related python virtual environment
```

(c) Off-chain: fingerprinting matching in the Python virtual environment

```
>>> if (isValidFingerprint(documentFingerprint)):
    exists = matchFingerprintWithDatabase(documentFingerprint)
>>> if(exists):
   documentName = getDocumentByFingerprint(documentFingerprint)
   transfer(from=ADMIN_ADDRESS, to=SMART_CONTRACT_ADDRESS,
args=["Right", True, documentName])
>>>else:
     addDocument(documentName, documentFingerprint)
     transfer(from=ADMIN_ADDRESS, to=SMART_CONTRACT_ADDRESS,
args=["Right", False, documentName, documentFingerprint])
```

(d) On-chain: user gets the updated Blockchain

```
#> if the transaction passes successfully:
# > the smart contract entry point is again available
# > a blockchain scanner notifies the client about the results
```

4.3. Fingerprinting Method and Database

In our study, we consider the camcorder fingerprinting method advanced in [9]. For scanned visual documents, the modules related to inter-frame synchronization are disabled.

The fingerprinting method is based on the 2D-DWT (2D Discrete Wavelet Transform) and on a mathematical decision rule for the detection of replicas.

The fingerprint per se is represented by a set of 2D-DWT coefficients of the query image. A previous in-depth statistical investigation on the 2D-DWT coefficients demonstrated not only the stationarity of such coefficients but also the stationarity of their modifications under computer-simulated near-duplicated modifications. Through its accurate representation of visual content, the wavelet transform grants the fingerprints the uniqueness property and limits the occurrences of false alarms (i.e., fingerprints extracted from different video content have to be different). The fingerprint matching is carried out, based on a repeated Rho test on correlation, which allows the detection of replicas, hence ensuring the

robustness property (i.e., fingerprints extracted from an original video sequence and its replicas should be similar in the sense of the considered similarity metric).

To evaluate the method performances, we created a 11,557 digital document database. The digital documents are JPEG images obtained from 25 PhD theses defended in France in various research fields (physics, biology, philosophy, . . .) accessible on [33]. To illustrate the process, some samples are shown in Figure 6; it can be noticed that the database features a very heterogeneous visual content, mixing text, numerical values, scientific graphical signs, graphical representations, and images. The database also features heterogeneity in terms of layout, font size, and language.

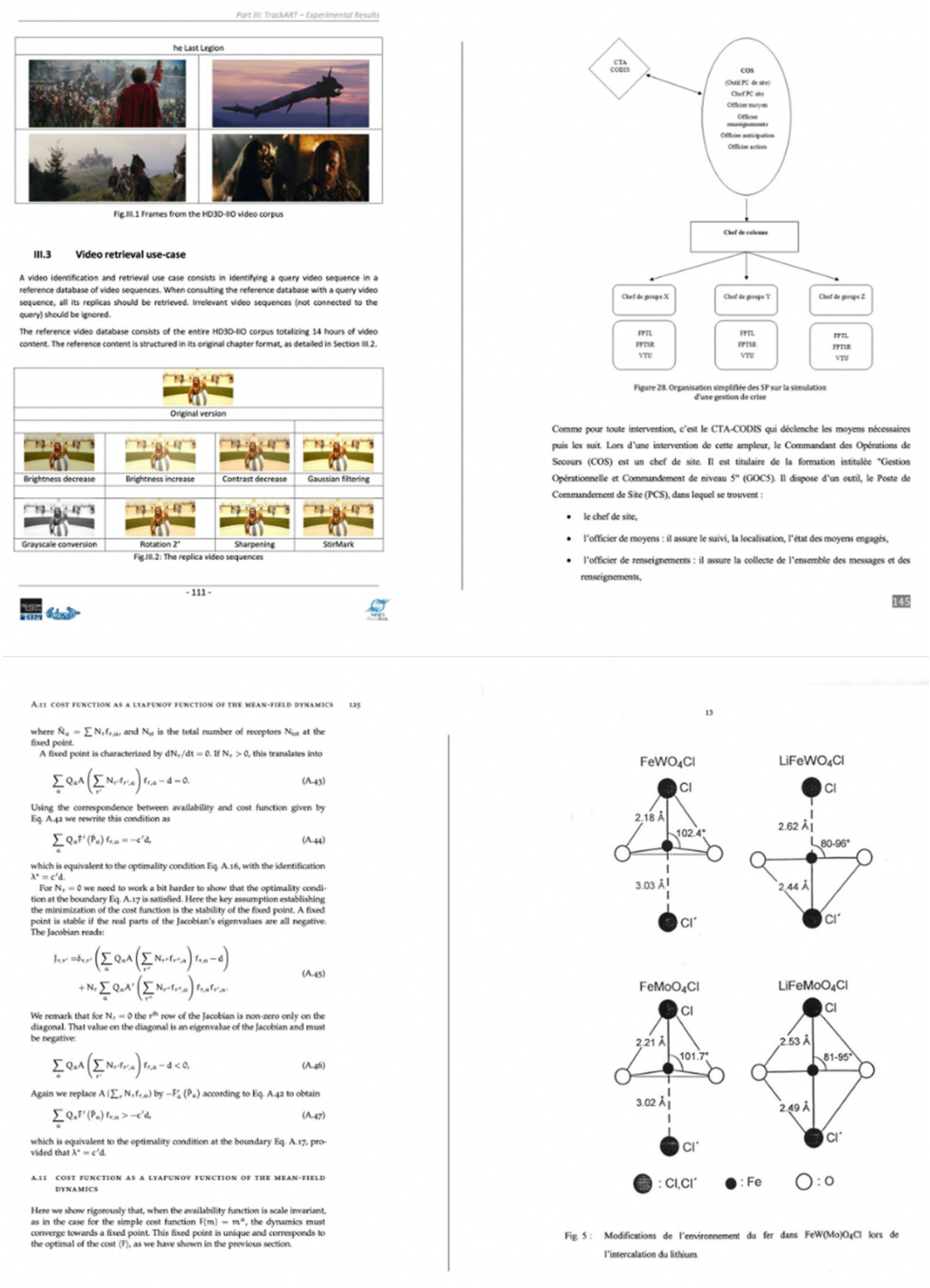

Figure 6. Original images sampled from database.

We also created a set of test images (attacked images). Each test image is composed of two parts extracted from two database images with different ratios ([50% image1, 50% image2], [67% image1, 33% image2], [75% image1, 25% image2]) as illustrated in Figure 7.

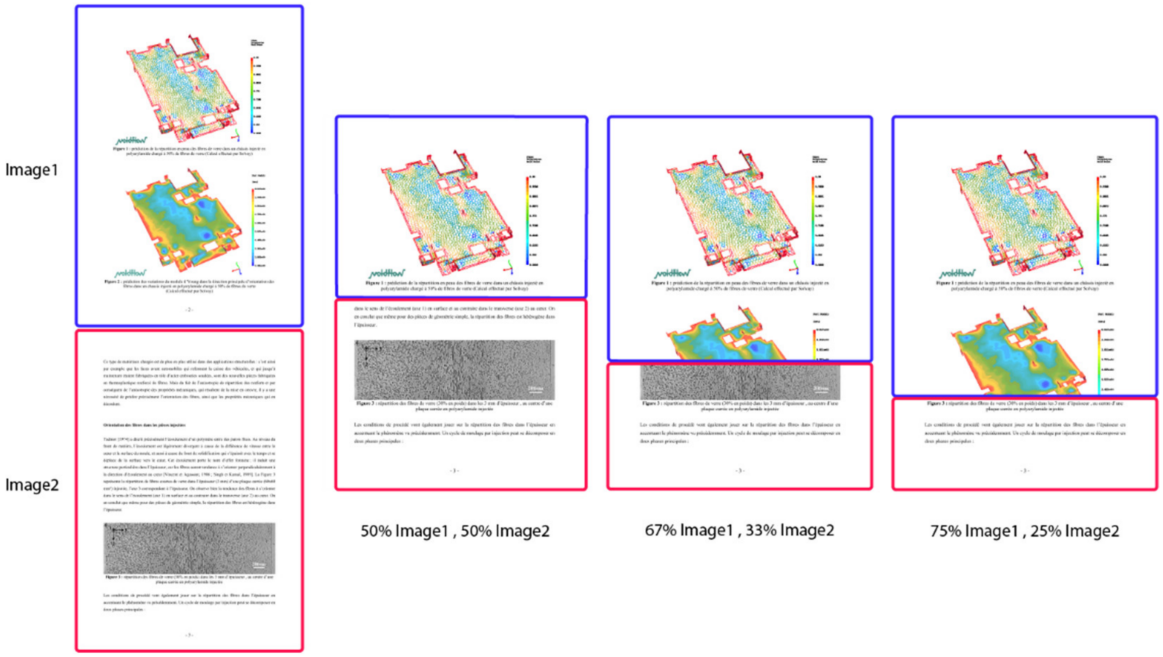

Figure 7. Attacked images sampled from database.

The resulting testing corpus is composed of 4 × 11,557 = 46,228 visual documents.

5. Experimental Results

In this section, we will describe the different experimental results obtained following the implementation of our application.

Figure 8 presents the memory consumption evolution while changing the number of deployed nodes: a quasi-linear behavior is thus brought to light. Yet, the experiments also brought to light that the maximum number of nodes to be deployed on a Raspberry Pi 3 is 9 (an error message is obtained when trying to deploy the 10th node). Note that this limitation is not derived from the memory: actually, after deploying the 9th node, only about 25% from the total memory is required (241.2 M out of 1 G). The values reported in Figure 8 are obtained as average values over 10 experiments; for each reported value, the 95% confidence limits are computed with a relative error lower than 5%.

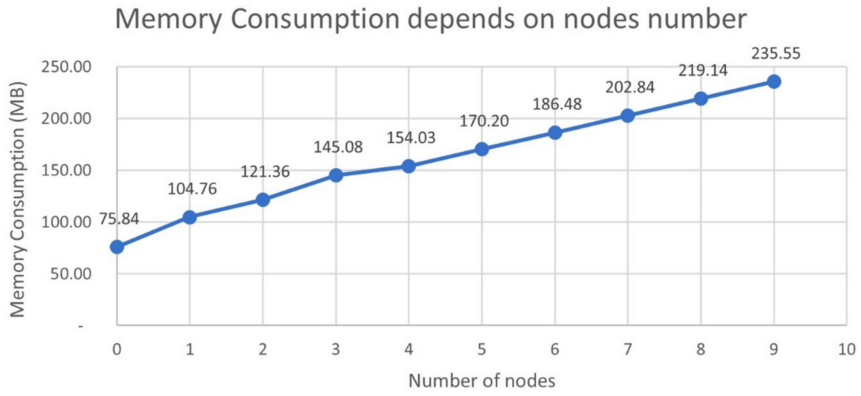

Figure 8. Memory consumption depends on nodes number.

These results demonstrate that the main limitation derives from the reduced computing resources: for more than 9 nodes, the maximal time imposed by the blockchain for performing basic operations cannot be met on a Raspberry Pi 3. The Raspberry Pi stops as illustrated in Figure 9.

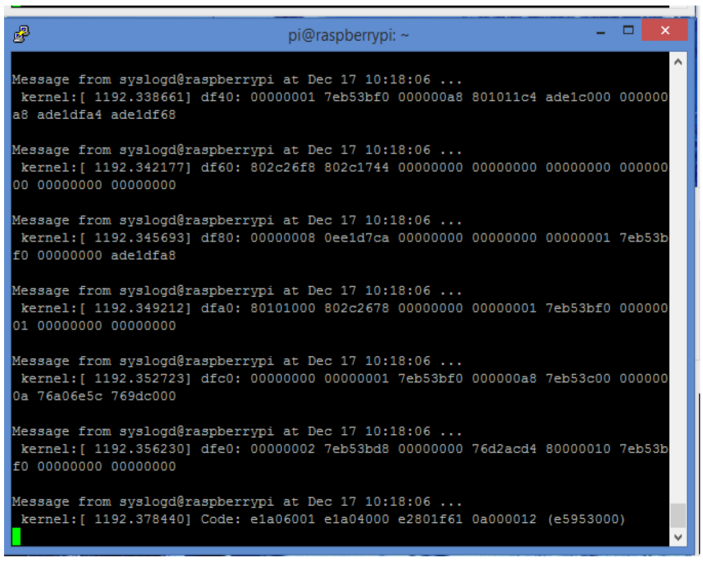

Figure 9. System crashes with more than nine nodes.

In the sequel, we deployed 5 Tezos nodes, so as to reach a trade-off between functional and non-functional needs.

Table 1 presents the execution time of the fingerprinting system (fingerprint extraction, matching algorithm, eventually the adding of new fingerprinting to the database) on a computer and on a Raspberry Pi, as a function of the visual document database size. By analyzing the results presented in Table 1, firstly, it can be noticed that the Raspberry Pi cannot process databases larger than 1000 documents (with 5 Tezos nodes running). When the database contains only one document, the execution time on Raspberry Pi is 37.7 times larger than the one on the PC. However, for databases containing between 10 and 1000 documents, the ratios of the Raspberry Pi execution time to the PC execution time vary between 7.5 and 9.8, with an average value of 9.13. For the two processing structures, these values also show the different weights of the initialization operations in the overall processing time.

Table 1. Overall execution time.

	PC	Raspberry Pi 3
1	7 ms	264 ms
10	42.2 ms	320 ms
50	69.1 ms	637 ms
100	103 ms	960 ms
500	375 ms	3.65 s
1000	704 ms	6.92 s
5000	3.41 s	—
10,000	6.78 s	—
11,557	7.77 s	—

The values reported in Table 1 are obtained as average values over 10 experiments. The underlying 95% confidence limits can be computed with relative errors lower than 10% for the case of a single visual document in the database and with relative errors lower than 2.5% for the rest of the cases. This difference can be explained by the fact that in the case of

a single document in the visual database, the random effect of the initialization conditions is higher than in the other cases.

Table 2 presents the execution time of Tezos Blockchain operations (Bake, Coin transfert, Typecheck, and Smart contract deployment) when considering 5 Tezos nodes running, on a PC and on a Raspberry Pi, respectively. The values reported in Table 2 are obtained as an average over 10 successive experiments; their corresponding 95% confidence limits can be computed with relative errors lower than 2.5%. For the four types of operations, the ratios of the Raspberry Pi execution time to the PC execution time is 2.23, 3.63, 6.27 and 3.95, respectively. It can be noticed that the largest ratio corresponds to the execution of the Typecheck operation: this result can be explained by the inner complexity of such a command related to the formal verification of the Smart contract.

Table 2. Tezos basic operations execution time.

	PC	Raspberry Pi 3
Bake	0.994 s	2.216 s
Coin transfer	1.261 s	4.575 s
Typecheck	0.505 s	3.167 s
Deploy a Smart contract	1.836 s	7.263 s

By comparing the results reported in Table 1 to those reported in Table 2, it can be noticed that fingerprinting-related operations induce a higher ratio of execution duration between Raspberry Pi and PC; this result can be explained by the fact that the Raspberry Pi considered in the experiments has a quad core CPU and our software implementation features an inner-parallelism designed to match the Blockchain operation peculiarities.

Table 3 presents the execution time of the digital document tracking system measured from the end-user perspective. It presents the time between the document sent and the result reception (however, it does not include the time required for adding new documents in the database). Table 3 shows that the Raspberry Pi execution time is 40% higher than using a classical PC solution (value computed with 95% relative error lower than 5%).

Table 3. Tezos basic operations execution time (for a 1000 fingerprints database).

	PC	Raspberry Pi
Time	13.6 s	23.7 s

Note that the results presented in Tables 1–3 are meant to illustrate the performance reductions when deploying the related operation on Raspberry Pi, as compared to a PC execution. They show that there is no applicative need for using a PC, the Raspberry Pi device being enough in this respect.

6. Discussion

The paper establishes the proof-of-concepts for using on-chain/off-chain load balancing as an enabler for Blockchain deployment on lightweight computing resources. Accordingly, a Blockchain of an arbitrarily large number of nodes can be deployed over any combination of computing resources, from cloud servers and PCs to Raspberry Pi.

Although the illustrations correspond to the Tezos Blockchain and to a visual fingerprinting application, these should not be considered as strong applicative constraints, as explained here-after.

On the one hand, the Tezos Blockchain has many appealing conceptual and applicative features, such as reliability and formal verification, on-chain governance and self-amendment. Yet, the results reported in the present paper are generic with respect to these peculiarities and can be considered as representative for any PoS Blockchain. Moreover, our solution features horizontal scalability. Actually, any Blockchain has a decentralized architecture, where nodes and groups of nodes are hosted on different computing resources.

Hence, while our paper shows that up to 9 nodes can be hosted on a Raspberry Pi 3 device, real-life Blockchain solutions are expected to deploy their required number of nodes over a combination of multiple Raspberry Pi devices, PCs, cloud servers, etc., as illustrated in Figure 10.

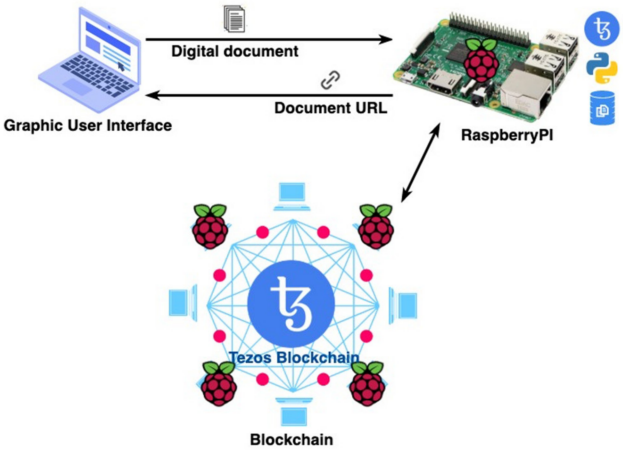

Figure 10. Nodes and groups of nodes can be hosted on different computing resources, from PCs to Raspberry Pi.

On the other hand, the overall solution advanced in the paper is not only (horizontally) scalable with respect to the Blockchain deployment but also with respect to the fingerprinting application, which has an O(nlog(n)) complexity [9].

In Figure 10, a PC is represented as a computing device for the graphic user interface; yet, it is not compulsory and can be replaced (without any impact in the experimental results) by any other device operated by an operating system featuring graphic user interface (as, for instance, a Raspberry Pi operated under Raspbian ver. September 2019). Moreover, in Figure 10, the database is graphically represented near a Raspberry Pi; yet, no particular constraint is applied to its storage in the Blockchain.

As a final remark, note that the applicative area of our solution is not restricted to fingerprinting. The solution for on-chain/off-chain load balancing can be mutatis-mutandi applied to any type of application, with limited impact in the codes of the modules presented in Figure 3:

- the Smart contract code is slightly modified so as to cope with the new applicative logic (yet the functions related to the on-chain/off-chain balancing are unchanged);
- the Secure REST Connector is not expected to suffer any modification;
- of course, the off-chain code will be completely changed so as to correspond to the new application.

7. Conclusions

Nowadays, companies and governments are taking the digital transformation challenge, and the means for reaching success in such a transformation are very different, belonging to technology, management, education, human acceptance, etc. One of the problems facing this transformation is digital documents tracking. While Blockchain is a priori the most appealing solution, digital documents proved themselves to be a challenging-use case that requires the coupling of Blockchain to visual fingerprinting.

With this paper, we establish a proof-of-concept for the coupling of a multimedia application (visual fingerprint) and Blockchain technologies and for the deployment of the underlying solution on a lightweight platform. To this end, we develop an on-chain/off-chain load balancing solution that is available (pending on paper acceptance) as open source. In this way, Smart contracts with very high computational complexity (the visual document tracking management, in our case) can be deployed on a Raspberry Pi structure.

The experimental results show that up to 9 Tezos nodes can be orchestrated on a Raspberry Pi 3 in order to ensure a visual fingerprinting task. The overall (end-to-end) computing time is 40% higher than using a classical PC solution. Note that all the values reported in our study are computed with 95% relative errors lower than 10%.

The illustrations correspond to the Tezos Blockchain and to a visual fingerprinting application: yet, the approach in itself is generic and does not depend on the blockchain/application peculiarities.

These results make it possible for a PoS Blockchain of an arbitrarily large number of nodes to be deployed over any combination of computing resources, from cloud servers and PCs to Raspberry Pi.

The short-term perspectives consider both software and hardware aspects. Firstly, stress tests and advanced security tests are required for the advanced on-chain/off-chain code balancing workflow. Secondly, hardware accelerators will be studied in order to reduce the overall execution time of the system (while observing pre-established energy consumption constraints).

Author Contributions: Conceptualization: M.M., G.M.; Methodology: M.M., M.A., G.M.; Software: M.A.; Experimental design and investigation: M.A., M.M., T.F.; Writing—original draft preparation: M.M., T.F., F.C.; Writing—review and editing: M.M., T.F., G.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly funded by the IPNG (le i-Parapheur Nouvelle Génération) French National Project. M. Allouche MS Thesis was founded by the PDMM (Protection des Données Multimedia sur Multicloud) project jointly funded by the SAMOVAR Laboratory at Telecom SudParis and LTCI Laboratory at Telecom Paris.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nakamoto, S. *Bitcoin Open-Source Implementation of P2P Currency*; P2P Foundation: Amsterdam, The Netherlands, 2009.
2. Available online: www.powercompare.co.uk/bitcoin/ (accessed on 10 May 2021).
3. Silva, T.B.; Morais, E.S.; Almeida, L.F.F.; Rosa Righi, R.; Alberti, A.M. Blockchain and Industry 4.0: Overview, Convergence, and Analysis. In *Blockchain Technology for Industry 4.0. Blockchain Technologies*; Rosa Righi, R., Alberti, A., Singh, M., Eds.; Springer: Singapore, 2020.
4. Issaoui, Y.; Khiat, A.; Bahnasse, A.; Ouajji, H. Smart logistics: Study of the application of Blockchain technology. *Procedia Comput. Sci.* **2019**, *160*, 266–271. [CrossRef]
5. Torky, M.; Hassanein, A.E. Integrating Blockchain and the internet of things in precision agriculture: Analysis, opportunities, and challenges. *Comput. Electron. Agric.* **2020**, *178*, 105476. [CrossRef]
6. Alonso, S.G.; Arambarri, J.; López-Coronado, M.; de la Torre Díez, I. Proposing New Blockchain Challenges in eHealth. *J. Med. Syst.* **2019**, *43*, 64. [CrossRef] [PubMed]
7. Chen, Q.; Srivastava, G.; Parizi, R.M.; Aloqaily, M.; Al Ridhawi, I. An incentive-aware Blockchain-based solution for internet of fake media things. *Inf. Process. Manag.* **2020**, *57*, 102370. [CrossRef]
8. Jiang, S.; Li, Y.; Lu, Q.; Hong, Y.; Dabo, G.; Xiong, Y.; Wang, S. Policy assessments for the carbon emission flows and sus-tainability of Bitcoin Blockchain operation in China. *Nat. Commun.* **2021**, *12*, 1938. [CrossRef]
9. Garboan, A.; Mitrea, M. Live camera recording robust video fingerprinting. *Multimed. Syst.* **2016**, *22*, 229–243. [CrossRef]
10. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. LSB: A Lightweight Scalable Blockchain for IoT security and anonymity. *J. Parallel Distrib. Comput.* **2019**, *134*, 180–197. [CrossRef]
11. Mingxiao, D.; Xiaofeng, M.; Zhe, Z.; Xiangwei, W.; Qijun, C. A review on consensus algorithm of Blockchain. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 2567–2572. [CrossRef]
12. Hellani, H.; Sliman, L.; Hassine, M.B.; Samhat, A.E.; Exposito, E.; Kmimech, M. Tangle the Blockchain: Toward IOTA and Blockchain Integration for IoT Environment. In *Hybrid In-Telligent Systems. HIS 2019*; Advances in Intelligent Systems and Computing; Abraham, A., Shandilya, S., Garcia-Hernandez, L., Varela, M., Eds.; Springer: Cham, Switzerland, 2021; Volume 1179. [CrossRef]
13. Frikha, T.; Chaabane, F.; Aouinti, N.; Cheikhrouhou, O.; Ben Amor, N.; Kerrouche, A. Implementation of Blockchain Con-sensus Algorithm on Embedded Architecture. *Secur. Commun. Netw.* **2021**, *2021*, 9918697. [CrossRef]

14. Frikha, T.; Chaari, A.; Chaabane, F.; Cheikhrouhou, O.; Zaguia, A. Healthcare and Fitness Data Management Using the IoT-Based Blockchain Platform. *J. Healthc. Eng.* **2021**, *2021*, 9978863. [CrossRef] [PubMed]
15. Durand, A.; Hébert, G.; Toumi, K.; Memmi, G.; Anceaume, E. The StakeCube Blockchain: Instantiation. In Proceedings of the 2020 Second International Conference on Blockchain Computing and Applications (BCCA), Antalya, Turkey, 2–5 November 2020.
16. Available online: https://entethalliance.org/wp-content/uploads/2018/05/EEA-Architecture-Stack-Spring-2018.pdf (accessed on 21 June 2021).
17. Su, X.; Huang, T.; Gao, W. Robust video fingerprinting based on visual attention regions. In Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, Taipei, Taiwan, 19–24 April 2009; Volume 109, pp. 1525–1528. [CrossRef]
18. Lee, S.; Yoo, C.D. Robust video fingerprinting for content-based video identification. *IEEE Trans. Circuits Syst. Video Technol.* **2008**, *18*, 938–988. [CrossRef]
19. Douze, M.; Gaidon, A.; Jegou, H.; Marszałek, M.; Schmid, C. INRIA-LEAR's Video Copy Detection System. TRECVID, 2008. Available online: https://hal.inria.fr/inria-00548664/ (accessed on 28 July 2021).
20. Haji Rassouliha, A.; Taberner, A.J.; Nash, M.P.; Nielsen, P.M.F. Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms. *Signal Process. Image Commun.* **2018**, *68*, 101–119. [CrossRef]
21. Available online: https://www.intel.com/content/www/us/en/programmable/products/fpga/features/stx-power-about.html (accessed on 12 May 2021).
22. Zynq. Zynq Ultrascale+ mpsoc. 2018. Available online: https://www.xilinx.com/support/documentation/user_guides/ug1213-zynq-migration-guide.pdf (accessed on 7 July 2021).
23. Irgens, P.; Bader, C.; Lé, T.; Saxena, D.; Ababei, C. An efficient and cost effective FPGA based implementation of the Viola-Jones face detection algorithm. *HardwareX* **2017**, *1*, 68–75. [CrossRef]
24. Gao, F.; Huang, Z.; Wang, S.; Ji, X. Optimized parallel implementation of face detection based on embedded heterogeneous many-core architecture. *Int. J. Pattern Recognit. Artif. Intell.* **2017**, *31*, 1756011. [CrossRef]
25. Zhao, X.; Liang, X.; Zhao, C.; Tang, M.; Wang, J. Real-Time Multi-Scale Face Detector on Embedded Devices. *Sensors* **2019**, *19*, 2158. [CrossRef] [PubMed]
26. Raspberrypi. Raspberry Pi 3 Model b+. 2018. Available online: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ (accessed on 1 July 2021).
27. Dolas, P.R.; Ghogare, P.; Kshirsagar, A.; Khadke, V.; Bokefode, S. Face Detection and Recognition Using Raspberry Pi. In *Techno-Societal 2020*; Pawar, P.M., Balasubramaniam, R., Ronge, B.P., Salunkhe, S.B., Vibhute, A.S., Melinamath, B., Eds.; Springer: Cham, Switzerland, 2021. [CrossRef]
28. Laila, U.; Khan, M.A.; Shaikh, M.K.; bin Mazhar, S.A.; Mehboob, K. Comparative analysis for a real time face recognition system using Raspberry Pi. In Proceedings of the 2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (IC-SIMA), Putrajaya, Malaysia, 28–30 November 2017; pp. 1–4. [CrossRef]
29. Shah, A.A.; Zaidi, Z.A.; Chowdhry, B.S. Daudpoto Real time face detection/monitor using raspberry pi and MATLAB. In Proceedings of the 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT), Baku, Azerbaijan, 12–14 October 2016; pp. 1–4. [CrossRef]
30. Tezos. Available online: https://Tezos.com/ (accessed on 11 May 2021).
31. "What Is Tezos," Blockgeeks. Available online: https://blockgeeks.com/guides/what-is-Tezos/ (accessed on 11 May 2021).
32. Goodman, L. Tezos—A Self-Amending Crypto-Ledger White Paper. 2014. Available online: https://academy.bit2me.com/wp-content/uploads/2021/04/tezos-whitepaper.pdf (accessed on 28 July 2021).
33. Investopedia. Available online: https://www.investopedia.com/terms/b/blockchain.asp (accessed on 19 December 2019).