Elastic Smart Contracts in Blockchains

Schahram Dustdar, *Fellow, IEEE*, Pablo Fernández, José María García, and Antonio Ruiz-Cortés, *Member, IEEE*

***Abstract*—In this paper, we deal with questions related to blockchains in complex Internet of Things (IoT)-based ecosystems. Such ecosystems are typically composed of IoT devices, edge devices, cloud computing software services, as well as people, who are decision makers in scenarios such as smart cities. Many decisions related to analytics can be based on data coming from IoT sensors, software services, and people. However, they are typically based on different levels of abstraction and granularity. This poses a number of challenges when multiple blockchains are used together with smart contracts. This work proposes to apply our concept of elasticity to smart contracts and thereby enabling analytics in and between multiple blockchains in the context of IoT. We propose a reference architecture for Elastic Smart Contracts and evaluate the approach in a smart city scenario, discussing the benefits in terms of performance and self-adaptability of our solution.**

***Index Terms*—Blockchain, elasticity, Internet of Things (IoT), smart cities, smart contracts, virtual chains.**

I. Introduction

CITIES are complex ecosystems, and their effective and efficient functioning has enormous impact on the quality of life of their citizens and society as a whole. However, building smart cities is probably one of the most difficult challenges our society faces today. Among the variety of problems that need to be solved, the question of how to leverage existing ICT technologies to develop foundations for smart city analytics in a transparent and trustworthy form greatly concerns all stakeholders in today's smart cities.

As of today, we have observed several technologies enabling the connection between social and technical subsystems for smarter city analytics. A huge number of Internet of Things (IoT) devices as well as human participation have been introduced to provide various types of data about urban mobility and transportation systems, electricity grid, smart buildings, manufacturing, intelligent logistics systems, and critical infrastructures. Cloud systems have been introduced and used to store and analyze these streaming, things-based, and social big (in terms of "volume, variety, velocity, and veracity") data through complex middleware for various analytics needed for the operation and optimization of cities. Human capabilities have been invoked in the loop to design and monitor cities together with software. All of these data, analytics capabilities, and domain knowledge in smart cities involves a large number of stakeholders, ranging from individual citizens to corporations, including also government agencies for both vertical and horizontal problems (such as energy consumption analytics or human mobility analytics). In this view, one needs to understand that analytics of smart cities are far from just "big data analytics" and IoT data analytics. Smart cities analytics have an inherent ecosystem requirement, leading to different paradigm shifts in big data analytics from transactions to ecosystem perspectives as well as in the involvement of multiple, not necessarily trusted stakeholders besides ICT sensors, networks and analytics.

Key city analytics often require data, analytics, and capabilities from both vertical and logical domains (e.g., related to energy consumption) in a complex ecosystem of things, software services, and people which results in multiple stakeholders, with varying trustworthiness degrees. Complexities in these analytics can be viewed by these stakeholders from different angles: *i) physical (space) view*: City analytics can be carried out for a single block, a street, or a house; *ii) logical domain view*: City analytics are needed for various vertical domains (e.g., building management, intelligent transportation management, and infrastructure maintenance) and horizontal domains (e.g., energy policy and governance, social wellbeing, and urban planning); and *iii) time view*: City analytics can be performed at different time-scales, e.g., online (with near real-time streaming data), offline (with historical data), as well as a combination of both near real-time and historical data, also considering accountability aspects. While current data gathering techniques are able to collect various types of data, state-of-the-art analytics techniques isolate data produced by technical systems (e.g., from sensors) and social systems (e.g., from people) and then centralize the data in centers (e.g., in clouds) to carry out analytics at centralized places (although utilizing parallel and distributed computing resources). Such approaches rely entirely on software capabilities to deal with big data captured through distributed hierarchical networks of computing elements.

In city analytics, data, information, knowledge, and computational capabilities from software services, things, and people are distributed in deep, interwoven distributed ICT

This work was partially supported by FEDER/Ministerio de Ciencia e Innovación – Agencia Estatal de Investigación under project HORATIO (RTI2018-101204-B-C21), by Junta de Andalucía under projects APOLO (US-1264651) and EKIPMENT-PLUS (P18-FR-2895), and by the TU Wien Research Cluster Smart CT. Recommended by Associate Editor Laurence T. Yang. *(Corresponding author: José María García.)*

S. Dustdar is with the Distributed Systems Group, Vienna University of Technology, Wien 1040, Austria (e-mail: dustdar@dsg.tuwien.ac.at).

P. Fernández, J. M. García, and A. Ruiz-Cortés are with the Smart Computer Systems Research and Engineering Lab, Research Institute of Computer Engineering, University of Sevilla, Sevilla 41012, Spain (e-mail: josemgarcia@us.es; pablofm@us.es; aruiz@us.es).

architectures. Therefore, state-of-the-art approaches are not adequate as they collect data at the edge of the city where things and people reside, bring the data to the root of the hierarchy (e.g., cloud), and perform analytics based on data provided by predefined settings. First, it does not support time-scale because fine-scale and coarse-scale data analytics are not interoperable, since either we miss a lot of data (in coarse-scale data) or we have to deal with lots of data (in fine-scale data). Second, this also makes the filtering and pre-processing data challenging for supporting complex logical domains, which must deal with different logical horizontal and vertical scales. Finally, we also have severe problems with physical scale: as most of the time we centralize data in one cloud data center so we do not have enough information to cover all physical spaces with sufficient quality to guarantee time-aware analytics, e.g., subjects to be analyzed change rapidly in physical world and we lack up-to-date information in the centralized computing environment.

We believe we need *flexible and elastic mechanisms to support city analytics by harnessing collective capabilities of things*, *people*, *and software to carry out timely*, *quality-aware*, *and elastic analytics spanning both horizontal and vertical domains*. Given the huge number of things, people, and software services easy to be found and utilized without the need of centralized control, we should investigate a fundamental paradigm shift in utilizing collective capabilities that are distributed across the city infrastructure to enable coordinated analytics in a flexible and elastic manner. Such analytics must be provided with adjustable quality of results for multiple stakeholders where complex, transparent, and trusted collaboration between things, software, and people are needed to understand and address past, current, and future problems of smart cities based on historical, current, and predicted data [1].

In this work, we discuss to what extent blockchain technologies are adequate to support complex analytics in these ecosystems. We first introduce a concrete motivating scenario in smart cities analytics (Section II) and analyze how existing approaches to smart contracts and virtual chains can be applied to carry out the relevant analytics (Section III). Our vision, described in Section IV, further develops the smart contract notion towards an elastic smart contract, which considers elasticity concerns, while providing a framework to horizontally and vertically integrate data and its associated analytics capabilities by promoting the idea of glue contracts. Then, we discuss the elasticity forces that drive our elasticity rationale in Section V. Section VI showcases our reference architecture to support elastic smart contracts, while Section VII evaluates our proposal based on a Hyperledger Fabric implementation. Finally, in Section VIII we conclude that our proposal will provide a comprehensive support for the different capabilities required in complex scenarios like smart cities.

II. Motivating Scenario

As depicted in Fig. 1, in this article we consider smart city infrastructures consisting of a) IoT sensors, b) edge devices, which perform computational tasks such as analytics tasks, c) more "powerful" edge servers (aka fog computing nodes), and d) cloud computing data centers as the fundamental architectural building blocks for sensing and processing IoT data in smart cities.

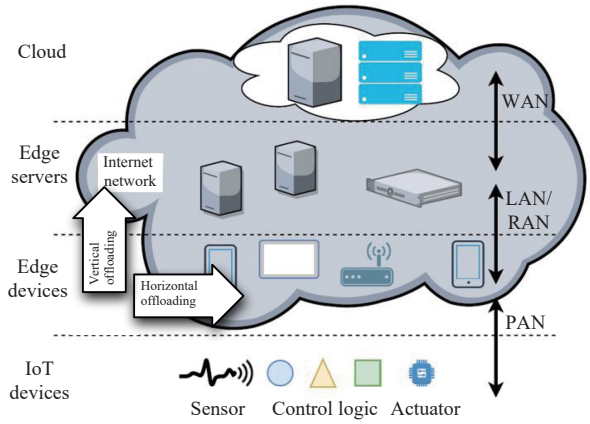

Fig. 1. Vertical offloading of analytics computation [2].

At the lowest level of the current smart city infrastructure, we see that data flows from the edges to the data center. From the infrastructure perspective, at the edge (e.g., buildings or districts) we can identify numerous capabilities offered by things, software services, and even people. At (and through) the data centers, several types of software services and people (from the crowds, professional groups, etc.) are available to perform data management and analysis. Although various types of infrastructures connecting people, IoT, and software services are distributed, current city analytics processes are mainly performed in the cloud using software services to provide results to humans. In principle, analytics processes can be carried out in multiple places within the city infrastructure by leveraging the collective capabilities of units of IoT, people, and software services. However, with today's techniques, such units cannot be collectively composed and provisioned on the fly for subsequent distribution throughout the city infrastructure. This prevents us from providing timely and elastic analytics to support non-functional concerns, such as cost, security, and privacy.

For complex problems, city analytics processes are logically divided into a set of sub-analytics processes that cover a set of concerns in distinct horizontal and vertical domains, as shown in Fig. 1. Computational tasks can be structured in a "vertical" way or in a "horizontal" way. Given the exemplified city analytics process for policy and regulation of sustainable environments, let us consider an analysis for a city block. Sub-analytics process concerns could be energy consumption of buildings and infrastructure, citizen wellbeing and opinions, environmental impacts of regulations, or incentive policies for green businesses, to name just a few. These sub-analytics processes belong to different vertical and horizontal domains and we need to correlate them and their results in order to understand how to create policy and how to regulate sustainable environments. In principle, such sub-analytics processes are also complex and some of them will be carried

out in the cloud (such as environmental impacts, and incentive policies) whereas others can be performed at the edge where things and people reside (e.g., building energy consumption, and citizen wellbeing and opinions). They also require different algorithms, data, and knowledge from different stakeholders. Among them, there are different ways to exchange analytics results and requests to ensure the final result of the city analytics to be delivered. To the best of our knowledge, state-of-the-art techniques just focus on centralized analytics for single domains. This leads to a severe problem for city analytics: As the scope of current analytics processes is limited to isolated domains and problems are either solved by software services or people, the results may not be adequate and substantial in the overall context of a city. We argue that smart city analytics must be researched from the perspective of ecosystems in which capabilities to contribute to analytics processes are based on hybrid resource types composed of software, people, and things, and where elasticity concerns may pose additional challenges to properly consider a dynamically changing number and type of those resources, with varying degree of quality and associated costs. Moreover, different stakeholders from multiple vertical and horizontal domains impose requirements on analytics processes due to the associated ecosystem of people, technology, and institutions.

Furthermore, in this scenario it is crucial that the shared data used to perform decentralized analytics in any dimension comes from trusted sources. However, considering the number of agents and stakeholders participating in a smart city ecosystem, trustworthiness cannot be assumed. Furthermore, certain stakeholders, such as public administrations, usually require transparency and tamper resistance to the data they use to analyze and provide services to other agents. For instance, a local administration may enact a contract with an external company to provide street cleaning services, using data from IoT sensors and possibly edge devices located on the streets to plan the optimal cleaning routes. Both the input data and the cleaning routes derived from its analysis should be publicly accessible in a transparent and immutable form, so that the local administration or even citizens can check whether the street cleaning company adheres to the contract in place and the quality level of the provided service, while providing flexibility and adaptability to changes in the ecosystem.

III. Related Work

A. Smart Contracts

In a complex scenario like the introduced before, a variety of stakeholders have to collaborate, sharing information between them and allowing each party to carry out analysis and provide decentralized services over the shared data. Trust issues become fundamental in this setting, since parties have to continuously agree on the validity of the data and services they need to integrate. Blockchain technologies are a natural fit, providing transparency and non-tampering to the data shared in a trustless network [3]. In addition to these features, privacy and rights management can be considered by using different blockchain implementations, ranging from permissioned blockchains [4] to specific solutions tailored to IoT-based ecosystems [5]–[7], including novel approaches to data management that focus on trust and privacy preservation [1], [8].

Since the introduction of smart contracts [9], blockchains have evolved from mere distributed digital ledgers to distributed computing platforms that can include not only an immutable data repository, but also logical and behavioral information to automatically rule the relationships between stakeholders. Thus, smart contracts can encode functionality needed to provide additional services on top of the data registered in the blockchain [10]. These contracts essentially aggregate some data under certain conditions that will trigger its execution. Although the data used within the contract logic is mostly obtained from the blockchain where the contract is deployed, oftentimes there is a need to consider external data (commonly referred as off-chain data). In order to retain the trustless characteristic of blockchains, an additional agent, namely an oracle, needs to provide the external data in a secured, trusted form [11].

B. Virtual Chains

Furthermore, there are scenarios where there is a need to separate nodes and information between different levels, as in our motivating scenario (see Section II). Virtual blockchains provide means to implement specific functionality on top of existing blockchains [12] by introducing an abstraction layer, so that the different application nodes subscribing to the virtual chain will access data and execute smart contracts tailored to their characteristics, while using a single blockchain as the backbone for recording every transaction within the whole system. Thus, multiple virtual blockchains (or virtual chains for short) comprising the different levels discussed in our motivating scenario can be deployed and integrated using this approach. However, sharing data between different virtual chains and from off-chain sources still needs the introduction of oracles, which could be just rights management systems in case of internal oracles allowing data access between virtual chains deployed on the same regular blockchain.

C. Elasticity

As the complexity of the systems grows, the need to adapt to variable flows of information and constraints to develop appropriate outcomes represents an important challenge. To this concern, elasticity [13] is presented as the capabilities to react and accommodate changes in the environment with an autonomous mechanism. In [14], authors provide a formal model of elasticity as a three-dimensional space involving resources, quality, and cost aspects that provide the appropriate framework to define and analyze the elasticity properties of an information system that will be used as a starting point of our conceptual proposal.

IV. Conceptual Proposal

Smart contracts represent an appropriate framework to

develop a computational mechanism combining data off-chain with the one present in the blockchain. However, in order to address the analytical challenges discussed in the motivational scenario, the framework should be extended to support a variable and multilevel nature of the actors involved. Specifically, in this section, we outline how the elasticity and integration aspects are fundamental cornerstones to build an appropriate smart contract ecosystem to develop more capable blockchains for complex scenarios such as a smart city. In this context, this conceptual proposal outlines the key design principles of the implemented framework described in Section VI.

A. Integration Concerns

Separating the information needs in different levels allows organizations to focus on their interests, while regulatory bodies can grant access to those organizations only to specific data. In this context, from a blockchain perspective there are multiple architectural alternatives to implement the level stratification, which can be characterized by analyzing three aspects:

1) Granularity: Several mapping options could be defined to assign a given blockchain to a single level (fine granularity) or multiple levels (coarse granularity). In addition, there could be some scenarios where the same levels are composed of multiple different blockchains.

2) Accessibility: From this perspective, we refer to the capability to analyze the blockchain content by different agents; i.e., the blockchain represents an open system (public) to any agent or a closed system (private or permissioned) to certain agents.

3) *Deployment model:* In this context, we address the logical implementation and deployment of the chain, i.e., existing blockchains that are implemented over a specific technical protocol or virtual chains that are materialized inside a regular existing blockchain.

Consequently, we can have a wide variety of modelling choices for a given scenario. As an example, Fig. 2 depicts a particular abstract scenario showing several options, namely Level 1 with two fine grain blockchains (one private and one public), Level *N* with one fine grain public blockchain that contains two virtual chains, and a coarse grain private blockchain that spans over all levels and contains a virtual chain for each level. From an analytics perspective, since smart contracts are meant to be executed in the context of a single blockchain, we envision the need for different cross-chain integration mechanisms (as exemplified in Fig. 2) depending on three factors: Whether integration is done between regular blockchains (Examples labeled with 1 and 2) or virtual chains (3 and 4); between chains in the same level (1 and 3) or different level (2 and 4); or between the same accessibility context (3 and 4) or between a public and a private chain (1 and 2). Taking these challenges into account, we claim the need for a special kind of smart contracts, coined as glue contracts, with the special responsibility of making data available across two different chains (virtual or regular) corresponding to the same level (horizontal integration) or different levels (vertical integrations).

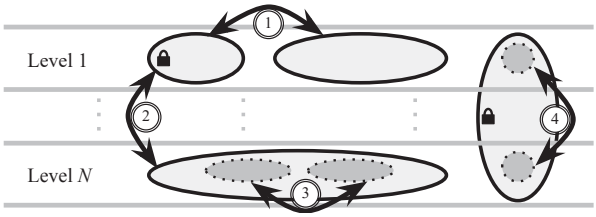

Fig. 2. Integration points between blockchains.

In this context, it is important to highlight that the discussed integration alternatives would represent different types of glue contracts. Thus, in order to integrate two different chains, a possible solution could make use of oracles in order to maintain the trust level of the whole ecosystem. In this particular case, the software oracles are just simple gateways to the accessed blockchains which do not need to add an additional trust method to the already trusted data from the accessed blockchains. Another example of mechanism used by glue contracts to address an integration between accessibility contexts could be the usage of IPFS[1] as the intermediary persistence area for data. In the case of cross-level (or vertical) integrations, glue contracts would be in charge of aggregating the data from inferior levels into new kind of information for higher levels. Furthermore, glue contracts need to address possible divergences between blockchain implementations and protocols of chains to be integrated. There exist alternatives to reconcile these divergences when dealing with crypto currencies [15], [16] that could be extended to allow dealing with complex asset integration.

B. Elasticity Concerns

Following the model presented in [14], our proposal takes into account the elasticity concerns to allow stakeholders to dynamically reconfigure the integration between levels, depending on the horizontal and/or vertical offloading needs (i.e., contract execution), by leveraging elasticity for analytical and glue contracts, correspondingly.

Specifically, in order to incorporate the elasticity dimensions in smart contracts, we need to provide means to elastically define resources, quality properties, and costs associated with a particular contract. To this end, we propose to add an abstraction layer to current smart contracts which will define the elasticity policies for a particular contract. Therefore, executing a so called *elastic smart contract* will transparently consider elasticity aspects on top of the actual functionality provided by the contract. Furthermore, stakeholders should consider executions costs for contracts (e.g., gas for Ethereum smart contracts) as well as infrastructural costs of the blockchains to plan the actual architecture of chains in levels. To this end, a decentralized market of agents [17] would allow the dynamic reconfiguration of the ecosystem taking cost information into account. In order to develop these implications, in Section V

[1] https://ipfs.io/

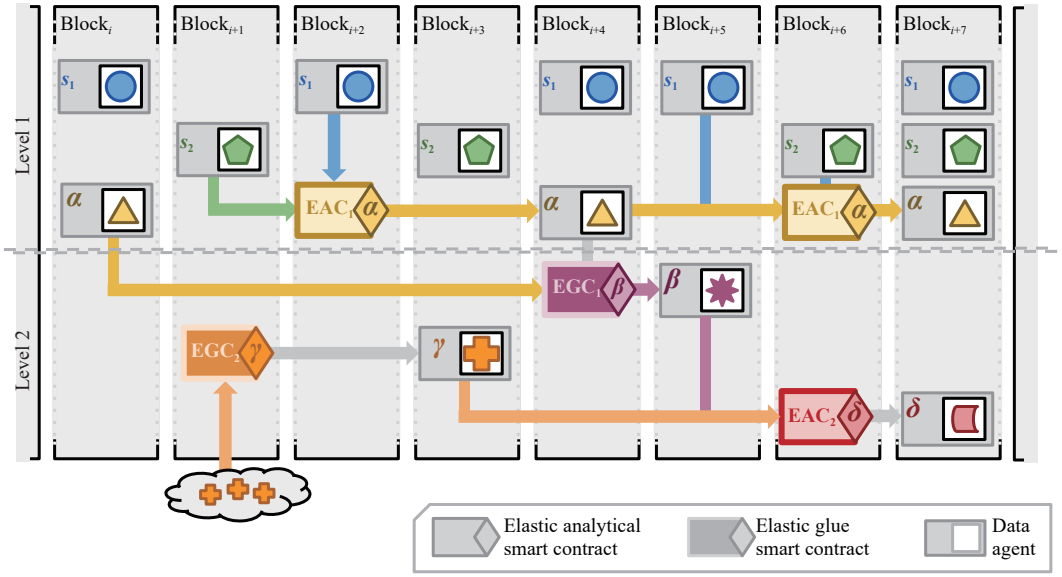

Fig. 3. Visionary use case with multiple integration points.

we detail the different elasticity forces derived from the nature of the blockchain paradigm.

C. Visionary Use Case

To exemplify the applicability of the proposal we outline a supporting architecture grounded on the current capabilities of blockchain technological state of the art. In such a context, in the current evolution state of the technology towards richer ecosystems, we expect continuous improvements and revisions of the conceptual frameworks presented. In this use case, (Fig. 3 shows a fragment of the envisioned blockchain) we can conceptualize an architecture of different virtual chains (composed of "virtual" blocks) that coexist in the same blockchain ecosystem (composed of "grounded" blocks) with smart contract capabilities (such as Ethereum). In such a framework, each grounded block would be a container for multiple virtual blocks that correspond to the different levels and contain either data or contracts related to that level. Specifically, in Fig. 3 we exemplify a fragment of the blockchain (Blocks i to (i+7)) including two levels (note that in a real scenario there potentially exists a higher number of levels). Inside Level 1 we can identify information generated by two agents (s_1 and s_2) and one elastic analytical smart contract (EAC_1) in charge of creating derived data from the activity in the level. Next, in Level 2, we can see two kinds of elastic glue contract (EGC). On the one hand, EGC_1 aggregates the information from Level 1 and incorporates the aggregation as new data in Level 2. On the other hand, EGC_2 (implemented as an oracle) imports data off-chain to the Level 2. Finally, EAC_2 analyzes the data of this level to create a new kind of information.

In this context, we can identify different examples of multiple interleaved analytics that can be mapped to the abstract blockchain fragment presented in Fig. 3 : from low-level analytics regulating small physical spaces that mainly involve sensor data, to high-level analytics involving other kind of data sources such as human actor decisions or off-chain census data. For the sake of clarity, we propose a simple example that would correspond with two low levels of analytics representing an adaptable urban lighting system:

Level 1 (street section) would represent a section of a street composed by a number of sensors and lights; concretely in the chain fragment depicted, agents s_1, s_2 could represent two presence sensors for a given road section that introduce their observations as data in the chain with different time resolution. The analytics contract EAC_1 would periodically perform an analysis over the sensors data to calculate a presence prediction (α) in the section; this analytical information would be used to actuate into adaptable street lights in the street section that switch on in the presence of cars, so they dynamically adapt their switch-off latency to the actual prediction.

Level 2 (street) the glue contract EGC_1 could aggregate the presence prediction of different sections calculated in Level 1 in order to create an estimation of the traffic flow in the street (β). In this level, the glue contract EGC_2 could include weather forecast as off-chain data (γ) so the analytics contract EAC_2 could calculate an estimation of the congestion risk (δ) in order to optimize the traffic lights rules for the given the street.

Furthermore, in a potential superior *Level N* we could leverage advanced use cases such as a new generation contract for waste management service that regulates the actual resource assignment algorithm based on the data harvested by the sensors; this could be implemented by a combination of elastic smart contracts using the analytics gathered and calculating the actual bills automatically, providing a completely transparent and non-tamper management procedure.

Examples of the three elasticity dimensions emerge from our use case: i) *Resources* range from the information providers that can correspond with things (e.g., sensors), software (e.g., government information systems) or people (e.g., an approval from a stakeholder); ii) Depending on the

type of resource, a taxonomy of *quality* aspects can be defined (such as resolution data in sensors, availability of the government information system or readiness of the stakeholder); iii) Finally, *costs* involved in the process can also be structured in terms of the resource type (e.g., energy cost of the sensor, infrastructure cost of the information system, or personnel costs). All these concerns would be taken into account to create the elasticity policies for each elastic contract; as an example in the use case, EAC_1 would have a policy to select the number of sensors (resources) filtered by a particular data frequency (quality) and constrained by a maximum number of gas used in the execution of the analytics (cost).

V. Elasticity Forces

As discussed in previous sections Elastic Smart Contracts are means to provide an appropriate analysis framework that is integrated into the blockchain taking into account resources, quality and cost considerations. Indeed, this elasticity need is grounded in the fact that analytics over a big set of data, performed by means of smart contracts in a blockchain, represent an actual challenge. In this section we analyze the different forces that would drive the elasticity rationale.

We can define two different kinds of data used as input for analytics (i.e., by means of smart contracts) in the blockchain paradigm. On the one hand, the paradigm provides a persistent, immutable and non-tampered way to store a set of transactions in the chain. On the other hand, for the sake of efficiency, in actual implementations of the paradigm, there are also other current (and mutable) data available in the ledger that can be used into the analytics (such as the *Asset* objects in Hyperledger Fabric). In such a context, although the access and modification of that mutable data is highly efficient, as the global size of that kind of data increases there could be a severe impact over the blockchain performance. Consequently, the performance implications of maintaining a big set of data, imposes a trade-off over the appropriate size of data that should be maintained for the analytics while keeping an appropriate blockchain performance.

We are going to discuss this trade-off proposing a number of dimensions that characterize the potential scenarios by means of three different perspectives.

First, in order to characterize the amount of data needed in a given scenario, we propose a general classification of analytics depending on the following two analytical requirements dimensions:

1) History: How much data should we retain to perform the analysis (i.e., how big should be the storage to keep the data).

2) Freshness: How current the data should be (i.e., how frequently should we harvest data to perform the analysis).

Taking both dimensions into considerations, Fig. 4 represents a quadrant that shows a wide range of types of analysis that are bounded in two extreme types: On the one hand (top-left), an analysis over a large set of historic data that is typically used for long term decisions that could have a subtle permanent effect such as the stop lights policies in our scenario; On the other hand (bottom-right), an analysis over a small set of fresh data that is typically used for short term

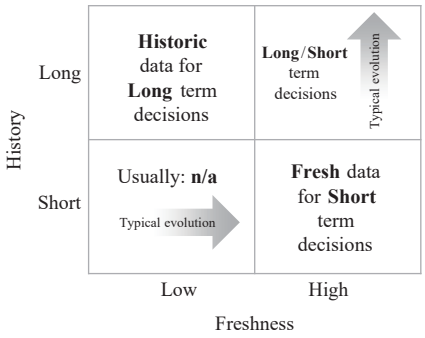

Fig. 4. Quadrant 1: History vs Freshness.

decisions that could potentially have a strong punctual impact such as switch on and off the street lights in our scenario. Complementary, there are two other hybrid possibilities (bottom-left and top-right) that could lean to those streams depending on the requirements in a particular scenario.

Second, depending on the actual scenario, in order to characterize its variability, we propose a global characterization of the dynamics of the analysis depending on the following two dimensions:

1) Data Density: In some scenarios we can have a fixed rate of data generated (e.g., an air quality sensor that takes one measure every second) or a variable flow of data depending on the situation (e.g., our example where the sensor in the street only generates the data if a car is detected).

2) Elasticity Requirements: These requirements could include any of the elasticity dimensions [14] (i.e., quality or cost considerations). In some scenarios, the requirements are fixed (e.g., maximum time for the analytics to be done), while in others there is a need for tuning the elasticity dynamically (e.g., a dynamic change on the threshold for the analysis execution cost).

Using those dimensions, in Fig. 5 , we show the quadrant that highlights the need for elasticity in a potential scenario. On the bottom-left case there is no need for elasticity, while on the rest there could be a suitable scenario for it. Conversely, in the most complex situation (top-right) the highest variability is present and, therefore, a potential elasticity mechanism will be highly required.

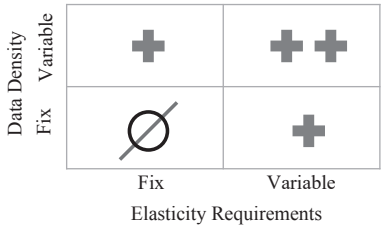

Fig. 5. Quadrant 2: Data Density vs Elasticity Requirements.

As a third perspective, we can combine the preexisting dimensions Data Density (i.e., Data generation rate) with the analytical requirements (i.e., Freshness and History) to propose a characterization of the stress levels that penalize the blockchain performance in a given situation derived from the amount of data that should be maintained in the ledger.

Specifically, in Fig. 6 , we can see how the lowest stress (bottom-left) covers situations with a low rate of data generation (i.e., low data density) while keeping a short list of historical data (i.e., short history) and a low update rate (i.e., low freshness). For instance, considering our scenario, in a situation with a small number of cars in the street (i.e., low car detections), only the last detection per sensor is stored, and the update frequency of the detections is set up every once in a while (e.g., once per hour). On the other extreme of the quadrant (top-right), we have a high density situation, with a frequent update and a long history. In our scenario, a related situation could need to cope with a high number of cars in the street, having to store a long list of detections (e.g., the last 50 detections per sensor), and to update very frequently those detections (e.g., once per second).

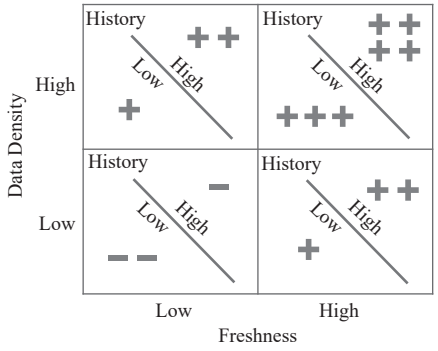

Fig. 6. Quadrant 3: Data Density vs Freshness/History.

VI. Realizing Elastic Smart Contracts

In order to support elasticity concerns for smart contracts execution and management, we devised the elastic smart contract reference architecture shown in Fig. 7 . We contextualize our proposal using the visionary use case introduced in Section IV-C, which we implemented using a Hyperledger Fabric [4] blockchain.

Specifically, Fig. 7 shows the key interactions by representing with solid lines the direct invocation or usage of a component (such as a smart contract) or resource (such as a configuration file) and by using the light green dashed line to represent the triggering of a component invocation based on a given event following the *publish-subscribe* pattern. Alternatively, the dashed dark green line represent the smart contract execution in the blockchain (Hyperledger Fabric in our implementation) triggered by an off-chain invocation of the appropriate API.

The main component of our proposal, called *Elasticity Orchestrator*, manages the execution of the blockchain system by monitoring the elasticity dimensions and calling the appropriate smart contracts according to the time and performance constraints of the particular scenario. These constraints are specified as elasticity rules (ER in Fig. 7), which define the lower and upper bounds admissible for the duration of the analysis execution, as well as the initial values for the history and freshness analytical requirements discussed in Section V, represented as history time window (HTW) and frequency of update (FUP) metrics in Fig. 7, correspondingly.

Within the boundaries of the *Elasticity Orchestrator*, we separate the responsibilities in two modules. On the one hand, the *Harvesting Manager* is responsible for managing the acquisition of input data from a set of sensors (or a data source in general) and updating the data stored in the blockchain that will be actually used to perform the appropriate analytics. Periodically, according to the current freshness required (i.e., FUP), the update management function obtains a set of detections $d_1, ..., d_n$ from the sensors being used and updates the data asset considering the current history time window (i.e., HTW). In order to do so, it submits a transaction to the Update Data smart contract, which will register the newly acquired data within the data asset that will serve as the input for the analysis smart contract, while removing the old data that falls behind the HTW.

In parallel to this process, the elasticity operation function performs the evaluation of the current status of the system with respect to the specified elasticity rules. Taking into consideration the average duration of the analysis execution (referenced as AD in Fig. 7), the elasticity operation periodically evaluates whether FUP and HTW should be changed to improve the expected performance of the elastic smart contract. This evaluation is performed by two smart contracts, named Evaluate Frequency and Evaluate History in Fig. 7, so that the evolution of the elasticity dimensions is registered and timestamped in the blockchain ledger.

On the other hand, the *Analysis Manager* module focuses on executing the actual analytics of the specific scenario supported by blockchains. Thus, the calculation management function is responsible for executing periodically (as stated in ER) the Analysis smart contract, which uses the Data asset updated by the harvesting manager as its input, and stores the analysis results (AR) in another Result asset located in the blockchain. The results from these executions are also collected by the metric gathering function, which aggregates both the AR and the generated performance statistics (PS), including the analysis duration (AD) used for the elasticity operation of the system. The Analysis smart contract communicates this information by emitting a New Result event, so that not only the metric gathering but also the elasticity operation can use the current metrics, especially AD, to dynamically adjust the elasticity dimensions FUP and HTW.

This reference architecture can be applied to any of the envisioned scenarios exemplified in Section IV-C. Although our proposed architecture is showcased and evaluated in the next section in the context of a single elastic analytical smart contract, the same approach can be replicated in more complex, multi-level scenarios, adapting our elasticity orchestrator to support elastic glue contracts, where the update management part supports the data integration from other blockchains or external oracles, while the calculation management handles the aggregation mechanism.

VII. Evaluation

In this section, we evaluate the benefits of our elastic smart contract reference architecture, analysing the effect of supporting elasticity concerns in analytical smart contracts.

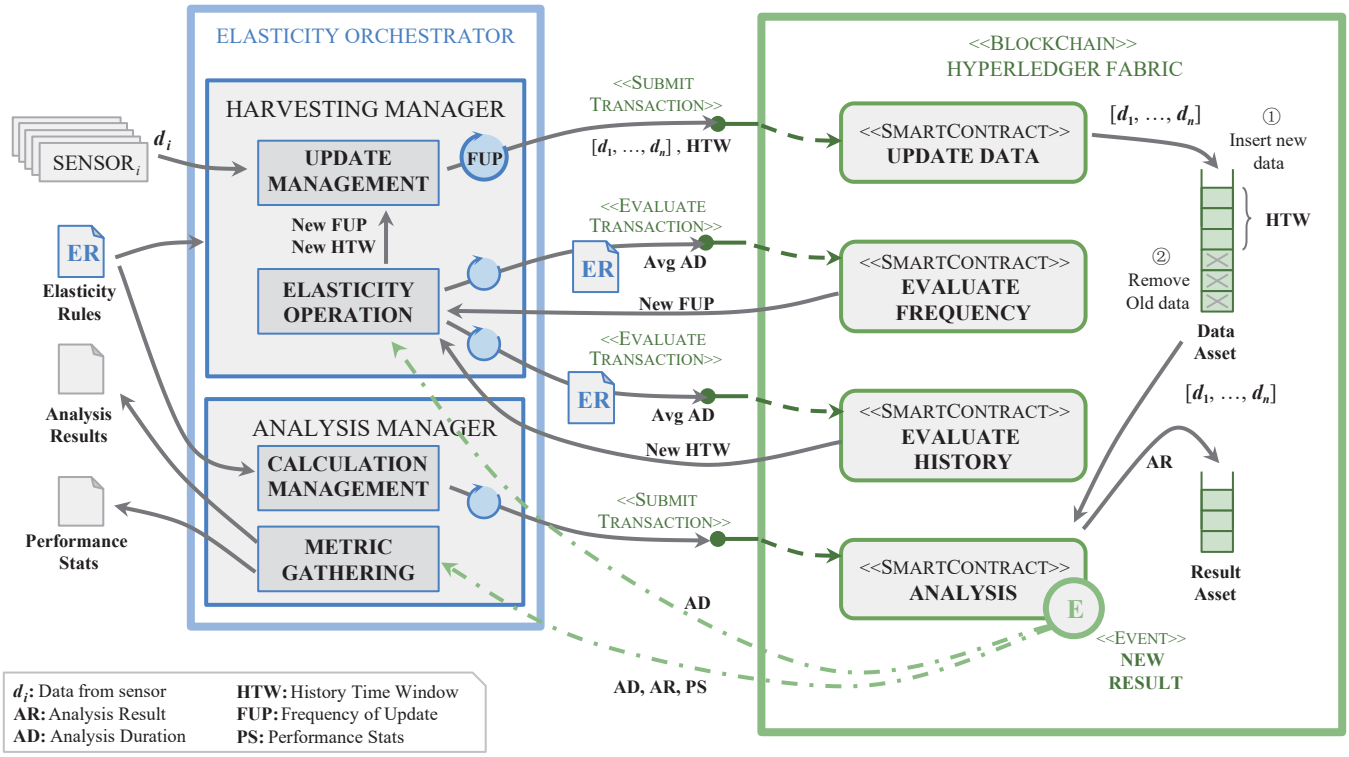

Fig. 7. Elastic smart contract architecture.

A. Implementation and Experimental Setup

In order to analyze the impact of the elasticity in our scenario, we developed a prototype of the elasticity orchestrator presented in Section VI The chosen blockchain framework was Hyperledger Fabric v2.1.0, deploying a test network using the default Raft consensus algorithm, and we implemented both the Elasticity Orchestrator and the smart contracts (*chaincodes* in Hyperledger Fabric terminology) using Node.js[2].

The use case scenario that we use to evaluate our proposal resembles the Level 1 example introduced in Section IV-C. Specifically, we simulate a variable number of sensors uniformly distributed alongside a street of variable length, and apply an analytic that calculates the average traffic flow of the street, expressed in average cars per second. We deployed the different smart contracts according to our proposed architecture in the three nodes that comprised our Hyperledger Fabric test network. Two of those nodes were associated with an organization, while the other was located in another organization. The consensus algorithm was configured so that any node may approve submitted transactions.

The experiments were executed in a computer equipped with an Intel Core i7-7700HQ 2.80 GHz CPU and 16 GB of RAM, running under an Ubuntu 18.04 operating system. We repeated each experiment three times and obtained average values for presenting the results[3].

B. Experimental Results

In general, all the experiments performed considered a street length of 1 km and 4 sensors distributed uniformly. The distribution of the input of cars on the street was also fixed, so that the analysis results obtained from calculating the car flow could be compared in terms of the analysis error. The simulation run for one hour for each experiment. With respect to the elasticity rules, we also fixed the following parameters:

1) The **elasticity range** for the analysis duration was set between 50 and 100 milliseconds, so any value outside this range would trigger the elasticity operation, modifying either the HTW or the FUP to maintain the analysis execution performance within the elasticity range.

2) The **frequency of analysis calculation** was set to 8 seconds, so that every 8 seconds the elastic orchestrator would try to execute the analysis smart contract.

3) The **elasticity evaluation frequency** was fixed at 5 seconds. This results in an evaluation of the system performance every 5 seconds, when the elasticity range is compared to the current average analysis duration.

As our baseline, we first evaluated how a regular blockchain implementation of the analytical contract without considering elasticity behaves. Fig. 8 summarizes the results of this experiment. First, the top diagram shows that, as the system warms up, the number of car detections stored in the blockchain increases up to 6000 detections. We had to impose this limit to the experiment because the blockchain starts to return consensus failures in our setup due to database congestion. Nevertheless, the figure clearly shows that most of the executions (62.81%) performed worse than our upper bound for triggering elasticity (100 ms), with 1.34% of the executions lasting more than 150 ms.

The bottom diagram in Fig. 8 shows the analysis results (cars per second), comparing them with the real calculation

[2] The prototype implementation and experiments launcher can be found at https://doi.org/10.5281/zenodo.4095100

[3] The complete experimental results obtained can be found at https://doi.org/10.5281/zenodo.4267996

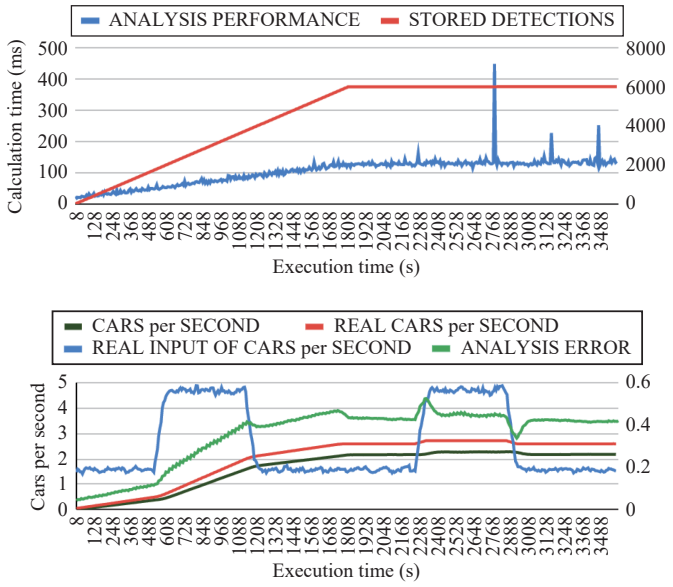

Fig. 8. No elasticity.

performed directly from the input data, so that we can show the analysis error between the real value for the traffic flow and the computed value obtained from the analytical smart contract execution. The blue line (real input of cars per second) represents the simulation pattern that we use for introducing sensor detections in our system. The analysis error increases especially after the density of car detections introduced increases, remaining around the average error value of 0.35 cars per second for the rest of the experiment.

For the second experiment, we introduced the previously described elasticity rules, and analysed how an elastic history window would affect the analysis performance. In this case, we fixed the FUP to 1 second, so that every second the elasticity orchestrator would update the detections stored in the data asset of the blockchain. The initial HTW was set at 1800 seconds (30 minutes). Fig. 9 shows the results we obtained. In this case, the number of stored detections (shown at the top diagram) that are used to calculate the traffic flow varies along the experiment duration, according to the elasticity operation of our system. At the beginning, as more cars enter the street, the number of stored detections increases. Once the HTW is filled with detections, since the analysis performance is below the lower bound, we increase the time window so that we can obtain a more accurate analytic without sacrificing performance.

However, by the 1250 s point of time, the performance degrades as it overcomes the upper bound of the elasticity range (100 ms). The bottom diagram of the figure shows how our elasticity orchestrator acts by decreasing the HTW, so that less data is used to compute the analysis. This results in a performance improvement, with the analysis duration now located within the elasticity range. For the rest of the experiment, the HTW remains largely the same, with some particular changes due to some spikes in the analysis performance. With respect to the analysis error, we can see in the middle diagram that the difference between the real calculation of cars per second remains low until the second rise on stored detections where there are worse values. However the average analysis error in this case is 0.33 cars per second, which even improves the error obtained without considering elasticity. But most importantly, the number of executions of the traffic flow analytic above the 100 ms threshold decreased dramatically to only 2% of them, while there was only one execution that spiked above 150 ms.

Finally, we performed an experiment where the elasticity is applied to the FUP. In this case, we fixed the HTW to store detections from the last 600 seconds, and started the simulation with a FUP of 1 second. Fig. 10 sums up the results of the experiment. When the simulation starts, as low density of detections are collected into the blockchain, the analysis performed is able to use all the data stored, so the analysis error is negligible. However, when more cars enter the street and hence more detections need to be analysed to properly calculate the traffic flow, the analysis error increases.

Since the analysis performance is contained or even below the elasticity range, our system starts to decrease the update frequency, so that the time between updates is lower. That causes an increase in the stored detections which in turn makes the elasticity orchestrator to increase the FUP so that less data is stored and the analysis performance improves again. This pattern is repeated every time the sensors detect more cars, which also causes the analysis error to fluctuate. Because of this, the average analysis error is higher in this case (0.37), and also the number of executions above the 100 ms threshold (25.73%), since the elasticity behaviour takes longer to affect the analysis performance.

C. Discussion

The most important conclusion that our evaluation results show is that adding elastic capabilities to smart contracts improves their performance, especially when applied to scenarios where there is a large amount of data generated, such as IoT-based analytics. Our system successfully maintains the analysis duration within the specified time limits, avoiding the saturation of the system by dynamically adapting the elasticity dimensions.

The effectiveness of our proposal relies on the limitation of history and freshness of the data, which in principle could affect the accuracy of the analytics performed. However, according to our experimental results, the analysis error is even more contained compared to the baseline, since our elasticity orchestrator allows the system to actually increase the considered history window or data freshness, hence improving the quality of the analytic results at some points. Nevertheless, when the elasticity operation reduces the history or freshness dimensions, the resulting analysis error increases, though on average we get similar results as the baseline approach.

Finally, comparing the two dimensions that we evaluated, the performance of the system is better when we applied an HTW-based elasticity in contrast to the FUP-based approach. We found that varying the history window the number of

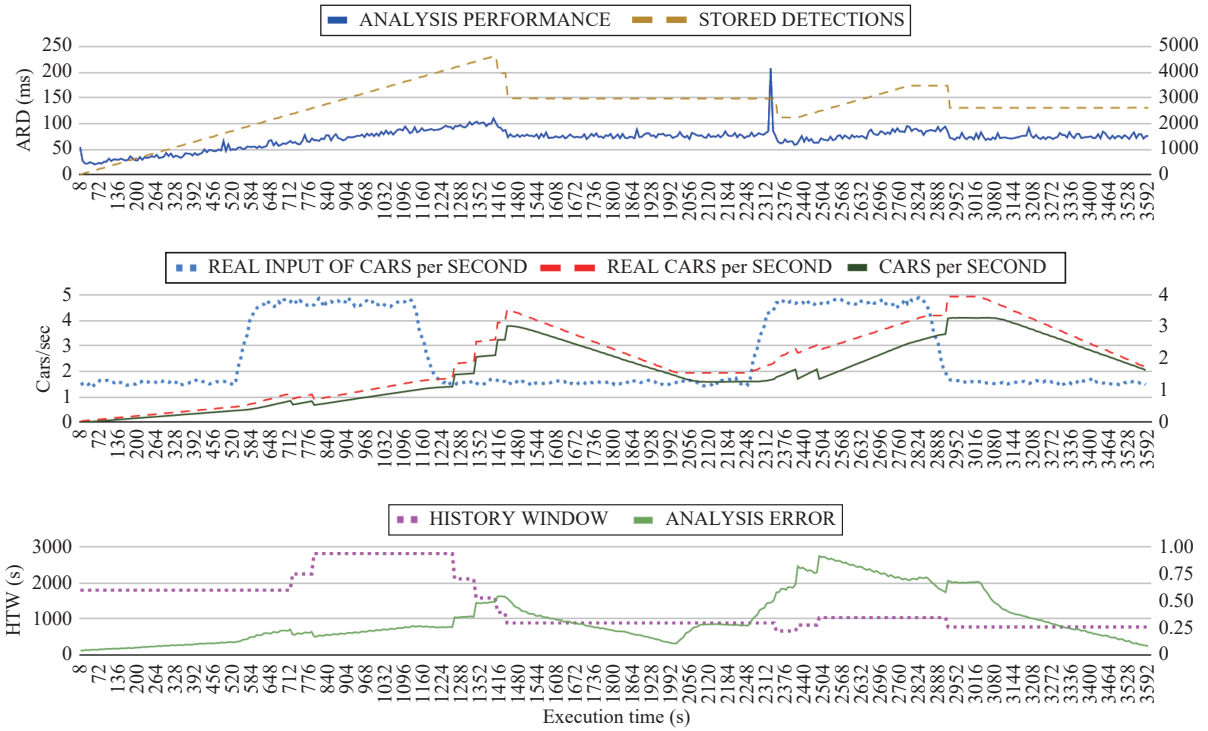

Fig. 9. HTW based Elasticity.

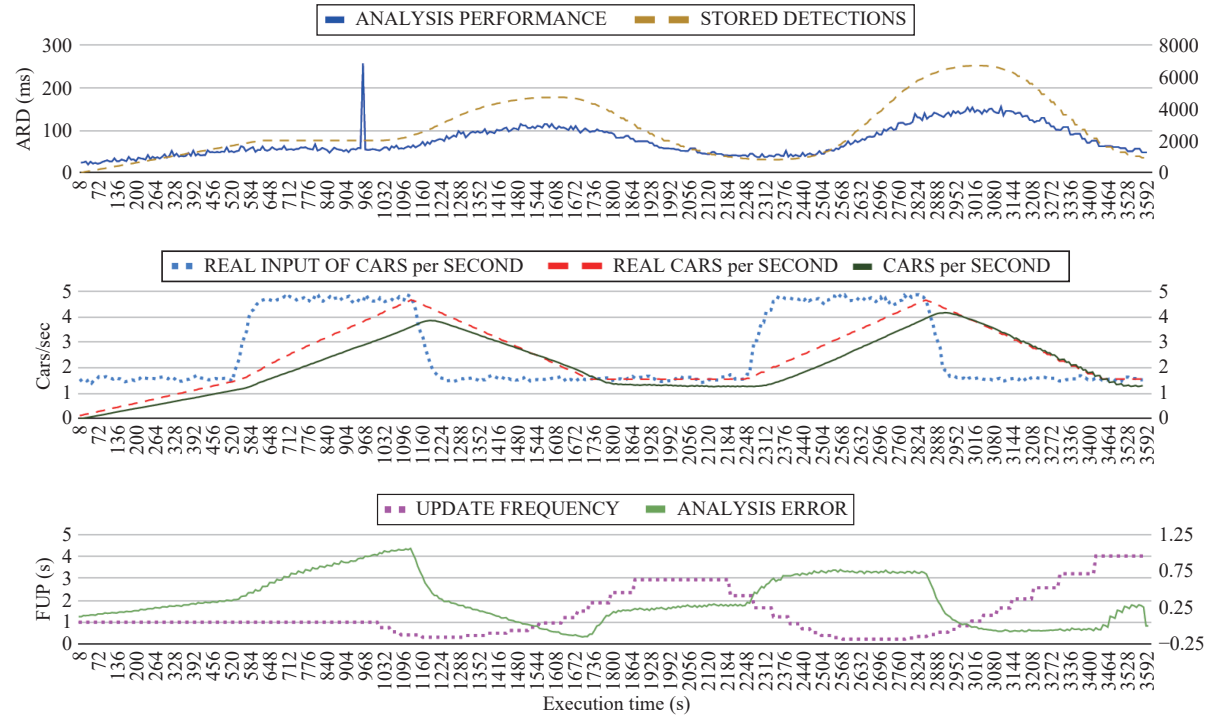

Fig. 10. FUP based Elasticity.

detections stored changes more abruptly, which in turn affects the system performance faster than changing the frequency of update of the data asset. The evaluation results show that the variation of the FUP is more gradual, which conduces to a more gradual effect on the analysis performance. Consequently FUP-based elasticity takes longer to stabilize the analysis duration, and suffers from a rebound effect, which causes a higher number of executions above the elasticity range, though still better than the baseline.

VIII. Conclusions

When facing complex scenarios as those that arise in smart cities, where transparency and accountability of the data and analytics are key goals, blockchains are a natural fit. However as these scenarios are typically composed by complex ecosystems of IoT sensors, edge devices, fog nodes, and cloud

data centers, the application of traditional blockchain technologies poses several challenges concerning elasticity and integration aspects, since the requirements for the analytics to be performed varies dynamically, not only in terms of resources needed, quality and cost aspects, but also in the dimensions of those resources. Thus, in order to support elasticity as well as horizontal and vertical integration, in this paper we introduce the concept of elastic and glue smart contracts.

The evolution of current blockchains towards supporting our envisioned elastic smart contracts needs the introduction of elasticity related information to the contracts logic. We propose an elasticity policy abstraction layer to extend the existing smart contracts introducing rules to account for variations in the three elasticity aspects (resources, quality and cost). Additionally, we characterize the different integration scenarios that can be applied to elastic smart contracts, exemplifying them in the context of smart cities and providing a reference architecture. Our vision is that using approaches such as virtual chains and adapting current elastic services frameworks, we can achieve a greater level of integration inside (and between) the various analytical levels while keeping a flexible reconfiguration of the architecture in case there is a need for vertical or horizontal offloading of computation.

As future work, we expect to apply the current approach to deal with more complex analytics such as the case of machine learning techniques where there is a need for reliable configuration parameters tuning. Moreover, we plan to apply the current framework to analyze real traffic information and to analyze the potentials of these analytics to drive real-time decisions (e.g., street light management) exploring the improvements in terms of operational expenses. Concerning the blockchain features used, we plan to extend the current framework to incorporate the cost of transactions (e.g., the gas in Ethereum) as another dimension of the elasticity requirements.

IX. Acknowledgments

Authors would like to thank Pablo García for his support on the prototype implementation and evaluation.

References

[1] M. Zhaofeng, W. Xiaochang, D. K. Jain, H. Khan, G. Hongmin, and W. Zhen, "A blockchain-based trusted data management scheme in edge computing," *IEEE Trans. Industrial Informatics* , vol. 16, no. 3, pp. 2013–2021, 2020.

[2] M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Dustdar, O. Scekic, T. Rausch, S. Nastic, S. Ristov, and T. Fahringer, "A deviceless edge computing approach for streaming IoT applications," *IEEE Internet Computing*, vol. 23, no. 1, pp. 37–45, 2019.

[3] M. Swan, *Blockchain*: *Blueprint for a New Economy*. O'Reilly Media, 2015.

[4] E. Androulaki, A. Barger, V. Bortnikov, S. Muralidharan, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Murthy, C. Ferris, G. Laventman, Y. Manevich, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, *EuroSys*. New York, USA: Association for Computing Machinery, Inc, 2018, pp. 1–15.

[5] A. Dorri, S. S. Kanhere, and R. Jurdak, " Towards an optimized blockchain for IoT," in *Proc. 2nd Int. Conf. Internet-of-Things Design and Implementation*. New York, USA: ACM, 2017, pp. 173–178.

[6] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2019.

[7] T. Jiang, H. Fang, and H. Wang, "Blockchain-based internet of vehicles: Distributed network architecture and performance analysis," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4640–4649, 2019.

[8] J. Feng, L. T. Yang, R. Zhang, and B. S. Gavuna, "Privacy preserving tucker train decomposition over blockchain-based encrypted industrial IoT data," *IEEE Trans. Industrial Informatics*, vol. 17, no. 7, pp. 4904–4913, 2021.

[9] V. Buterin, " A Next-Generation Smart Contract and Decentralized Application Platform," Ethereum.org, Tech. Rep., 2013. [Online]. Available: https://ethereum.org/en/whitepaper/

[10] X. Huang, D. Ye, R. Yu, and L. Shu, "Securing parked vehicle assisted fog computing with blockchain and optimal smart contract design," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 426–441, 2020.

[11] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, " The blockchain as a software connector," in *Proc. 13th Working IEEE/IFIP Conf. Software Architecture*, 2016, pp. 182–191.

[12] J. Nelson, M. Ali, R. Shea, and M. J. Freedman, " Extending existing blockchains with virtualchain," in *Proc. Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[13] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, " Elasticity in cloud computing: State of the art and research challenges," *IEEE Trans. Services Computing*, vol. 11, no. 2, pp. 430–447, 2018.

[14] G. Copil, D. Moldovan, H. L. Truong, and S. Dustdar, " RSYBL: A framework for specifying and controlling cloud services elasticity," *ACM Trans. Internet Technology*, vol. 16, no. 3, pp. 1–20, 8, 2016.

[15] M. Herlihy, " Atomic cross-chain swaps," in *Proc. Annu. ACM Symp. Principles of Distributed Computing*. New York, USA: Association for Computing Machinery, 2018, pp. 245–254.

[16] A. Hope-Bailie and S. Thomas, " Interledger: Creating a standard for payments," in *Proc. 25th Int. Conf. Companion on World Wide Web*. New York, USA: Association for Computing Machinery, 2016, pp. 281–282.

[17] J. M. García, P. Fernandez, A. Ruiz-Cortés, S. Dustdar, and M. Toro, "Edge and cloud pricing for the sharing economy," *IEEE Internet Computing*, vol. 21, no. 2, pp. 78–84, 3, 2017.

Schahram Dustdar (F' 16) is Full Professor of computer science heading the Research Division of Distributed Systems at the TU Wien, Austria. He is founding Co-Editor-in-Chief of the new *ACM Transactions on Internet of Things (ACM TIoT)* as well as Editor-in-Chief of *Computing* (Springer). He is an Associate Editor of *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*, as well as on the editorial board of *IEEE Internet Computing* and *IEEE Computer*. Dustdar is Recipient of the ACM Distinguished Scientist Award (2009), the IBM Faculty Award (2012), an Elected Member of the Academia Europaea: The Academy of Europe, where he is Chairman of the Informatics Section, as well as an IEEE Fellow.

Pablo Fernández received the Ph.D. on technology and software engineering from the University of Sevilla, Spain, in 2013. Since 2019 he has been an Associate Professor with the Applied Software Engineering Group, Smart Computer Systems Research and Engineering Lab, Research Institute of Computer Engineering, University of Sevilla, Spain. His current research is focused on the automated governance of organizations and infrastructures based on service level agreements and commitments.

José María García received the M.Sc. degree (Hons.) in computer science and software engineering from the University of Sevilla, Spain, in 2008, and the Ph.D. on technology and software engineering from the same university in 2012.

From 2006 to 2013, he was a Research Assistant with the University of Sevilla. From 2013 to 2015, he was a Postdoc Researcher and Assistant Professor with the Semantics Technology Institute of the University of Innsbruck, Austria. In 2015, he rejoined the University of Sevilla as Assistant Professor, and since 2019 he has been an Associate Professor with the Applied Software Engineering Group, Smart Computer Systems Research and Engineering Lab, Research Institute of Computer Engineering, University of Sevilla, Spain. His research focus is on blockchain, semantic web technologies, service-oriented architectures and linked data.

Antonio Ruiz-Cortés (M' 20) received the B.Sc. degree (Extraordinary Award) in computer science from the University of Sevilla, Spain, in 1992, and the M. Sc. and Ph. D. degrees (Extraordinary Award) on informatics engineering from the same university in 1996 and 2002, respectively.

From 1990 to 1998, he was in the Computer Industry. From 1996 to 1998, he was a Part Time Lecturer with the University of Huelva. In 1998 he joined the University of Sevilla as Full Time Lecturer. In 2004 he founded the Applied Software Engineering Group at the University of Sevilla. Since 2016 he has been a Full Professor of software and service engineering and since 2019 he heads the SCORE Lab at the University of Sevilla. His current research focuses on service-oriented computing, business process management, testing and software product lines. Prof. Ruiz-Cortés is Elected Member of the Academy of Europe and since 2018, President of the Spanish Society on Software Engineering (SISTEDES). He is also an Associate Editor of *Springer Computing and International Journal of Cooperative Information Systems*; and Recipient of the Most Influential Paper of SPLC (2017) and VAMOS Award (2020).