Blockchain-Based Decentralized Cloud/Fog Solutions: Challenges, Opportunities, and Standards

Rafael Brundo Uriarte and Rocco De Nicola

Abstract

Smart contracts and blockchain have the potential to change the current shape of cloud markets by enabling the development of completely decentralized cloud/fog solutions, which lower costs and enforce predictable results without requiring any intermediary. In this article, we survey three of these solutions, namely Golem, iExec, and SONM, compare them, and identify some of the problems they leave unsolved. Moreover, we consider existing standards for the development of interoperable decentralized cloud solutions that would allow such systems to compete with large providers and would prevent vendor lock-in. We believe that our study contributes to the evolution of cloud systems not only by pointing out incompatibilities among projects and possible solutions for research problems in the area, but also by reviewing the existing standards and suggesting new standardization opportunities.

Introduction

Smart contracts and blockchain are revolutionizing business by offering the possibility of removing intermediaries. Additionally, they have the potential to change the current shape of cloud/fog markets and lower entry barriers for such markets. In fact, despite the current efforts to advance interoperability between cloud offers, the cloud market has developed without standards and is restricted to a few providers [1] that have a dominant market position. This limitation has a significant impact on medium/small providers, who cannot easily enter the market; and on consumers, for whom changing provider is difficult since the service conditions are difficult to compare because every provider has its own vocabulary. Also, consumers need to verify data compatibility, and to consider transfer costs and applications adaptation, which together might lead to vendor lock-in.

Smart contracts and blockchain enable the creation of blockchain-based decentralized cloud solutions to face these problems. Smart contracts [2] are computer protocols intended to facilitate, verify, or enforce a contract; they are described in an executable language and enable execution of trusted transactions without third parties. Blockchain [3] is a viable solution to securely register transactions and to prevent fraud, even in completely decentralized systems. Together, they bring low costs and predictable results to cloud markets without resorting to intermediaries, with the exception of a trusted computing system. Blockchain-based decentralized clouds render the traditional long-lasting relationship between provider and consumer more dynamic by opening the market and simplifying the procedure for changing provider.

The development of blockchain-based solutions for cloud computing has only recently started and focuses on commercial targets. We survey three fresh projects,[1] namely Golem [4], iExec [5], and SONM [6]. These are not the only projects exploiting blockchain for handling distributed resources. There are others specifically designed to manage distributed storage,[2] and further ones[3] with similar aims. We consider the selected projects as the ones offering the most mature solutions, which are instrumental in singling out challenges for the research community and in defining standards that could guide future developments of interoperable decentralized cloud solutions.

Our study is aimed at bringing to light the potential of blockchain in the decentralized cloud domain and highlighting its current limitations. In fact, we believe that many of the design choices of the considered projects have been made on an *ad hoc* basis without systematic study and specific attention to standards. Our analysis could then be useful for pointing out incompatibilities, research challenges, and new standardization opportunities.

Blockchain-Based Decentralized Cloud Solutions

The three projects we consider (Golem, iExec, and SONM) rely on blockchain, and in our work, we focus on the technical aspects of their architectures. We would like to stress that information about the solutions adopted by these projects is sparse and difficult to find; they are ongoing projects with a high degree of dynamism, which use many communication channels, where only specific parts of the proposed solutions are considered. Thus, our presentation relies mostly on project white papers, documentation, blogs, and newsletters, and some changes may take place even before the actual implementations of what is described on the web.[4]

[1] **Disclaimer:** Our analysis is strictly technical and by no means consider the market cap, media impact, and so on. Also, we have no relation with any of these projects, and none of us holds any of their cryptocurrencies.

[2] https://filecoin.io/, https://storj.io/, https://safenetwork.org/, etc.

[3] https://www.iagon.com/, https://dadi.cloud/en, etc.

[4] In the blog of the Golem project, the complexities of the field are acknowledged, and it is stated that several points in the initial roadmap have been changed "not to get stuck with endless research." We believe that the other projects had to confront similar issues.

Digital Object Identifier: 10.1109/MCOMSTD.2018.1800020

The authors are with IMT School for Advanced Studies Lucca.

The three projects have their own coins/currency but share several features. Their common high-level architecture is reported in Fig. 1. Due to the high computational cost of the blockchain, smart contracts, payments, and reputation are managed in a separated network, called the *transaction network*, and services are executed as *off-the-chain computations* in a *side-chain network*. Tasks are executed by third-party resource providers (any user can provide resources and be rewarded), which communicate through a peer-to-peer (P2P) network. The side-chain is responsible for service execution, negotiation, and results verification.

Smart contracts are designed for deterministic environments, and they trade off immutability and trust for flexibility. This determinism is necessary since (at least) a significant part of the nodes of the transaction network executes the smart contracts to achieve consensus about the fulfillment of conditions. The side-chain is, however, non-deterministic, which can lead the nodes that execute the smart contracts to yield different results and thus prevent consensus. Due to this distinction between the transaction network and the side-chain, an interface, the so-called *oracle*, is necessary to select the "true" results among the provided ones, thus enforcing determinism in the transaction network.

Service provisioning is illustrated in Fig. 2. The consumer chooses or sends his/her own application and requests the execution of service. The transaction framework matches the request with an offer and creates a smart contract. The consumer sends the data to the scheduler, which forwards it to the distributed storage and requests the execution of the service to the available providers. The providers download the data necessary to run the services, execute them and send the results back to the verification module, which will either confirm that the result is correct or will request other providers to (re-)execute the service or some of its parts. Finally, the transaction framework transfers the payment to the providers, confirms the transaction and updates the reputation system.

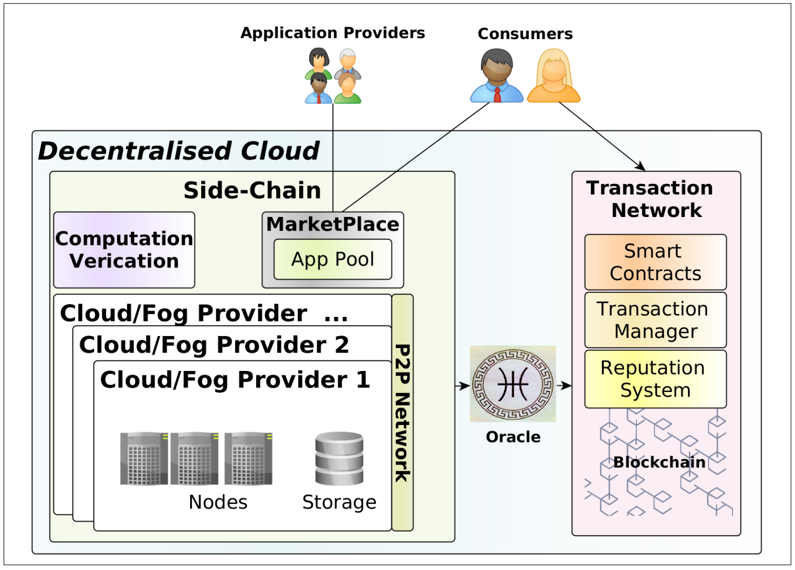

FIGURE 1. A high-level architecture of the decentralized cloud/fog solutions.

GOLEM NETWORK

Golem's [4] development began in 2014 and in November 2016 it raised \$8.6 million in a public Initial Coin Offering (ICO) from investors. The project is open source, and it is claimed that all cloud service models — infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) — will be covered. Despite this claim, neither in the white paper nor in the additional documentation could we find an example or a description of IaaS and PaaS services. Probably this choice was made due to the technological solutions adopted in the project, the larger target audience, and the simplicity of this service model for the target consumers.

Golem's architecture is presented in Fig. 3. The P2P network maintains the connection between nodes using end-to-end encryption elliptic curve. Providers offer one or more nodes, and services are executed directly on the node or in sandboxed environments. In this case, Dockers and virtual machines (VMs) are supported; data are shared and stored in the distributed storage module, which is based on DAT,[5] a data sharing protocol for P2P systems. Developers can create specific applications, using a *developer kit*, and templates in the *task definition framework*, to make their applications available in the *application registry* of the Golem network. The template contains the computation logic, the code to be executed, the specification of how to split services into tasks, and how to merge and verify results. If consumers want to run custom applications, they should define them using the task definition framework.

Payments are handled by the transaction network that also manages information about contributions to tasks and results. The transaction network relies on Ethereum for rewarding providers for computing tasks. It is also the base of the *reputation system* and of the *transaction framework* that registers and validate transactions. The reputation of providers increases if their computations end correctly, and decreases in case of errors (due, e.g., to hardware failure), no response, or wrong results attributed to malicious behaviors. Since registering transactions in Ethereum is costly, due to the potentially high number of tasks for each service, the Golem team developed a solution to prevent the need for third parties in regulated transactions.

Golem supports several types of payment, such as batch and nano-payments. The latter scheme was devised to keep network fees below 1 percent. Since a service can be divided into many tasks that can be executed by as many providers as the numbers of tasks, instead of splitting the payment and creating many transactions, which imply high fees, Golem uses a probabilistic payment scheme; only a few providers are paid in each transaction. A provider has a probability v_i/v to receive the payment for the service, where v_i is the payment due to him/her for the task he/she computed, and v is the total price of the service. This approach considerably reduces the transactions fees but does not guarantee fair payments to providers. However, the actual income approaches the expected one when a provider executes many tasks, which motivates providers to be active in the network.

[5] https://datproject.org/

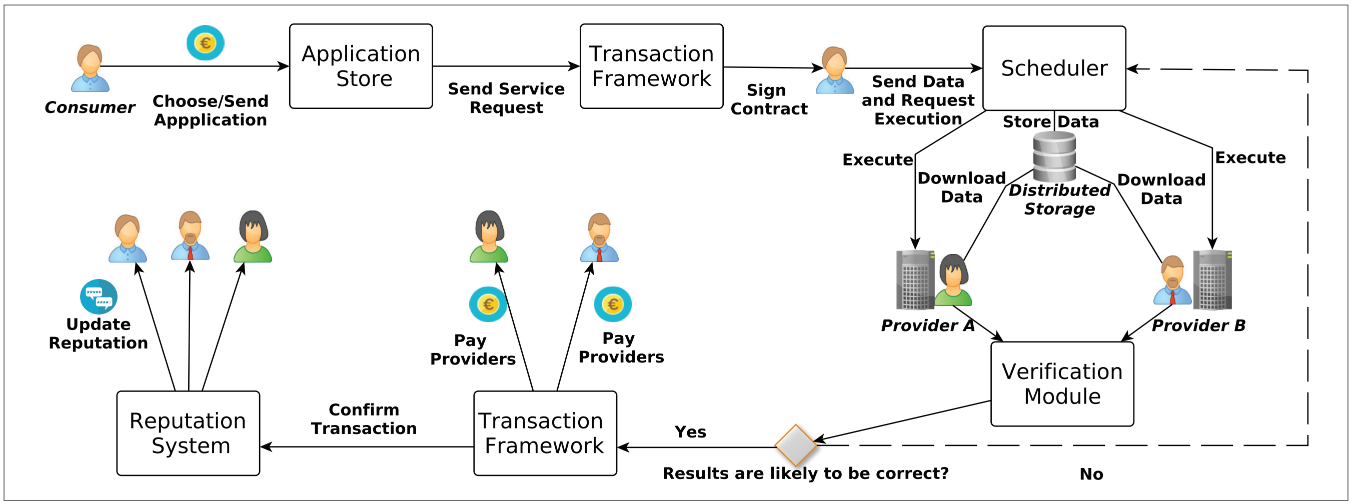

FIGURE 2. A standard service provision in decentralized cloud/fog.

To prevent malicious behavior, the *verification module* checks whether the tasks were really executed by using redundant computation, thus letting multiple providers execute the same tasks and comparing the results. Moreover, it performs simple correctness checking by using asymmetrically verifiable problems (where the result is cheap to verify but difficult to compute), redundant computing (executing small and random parts of the task and comparing the results), and log analysis.

iExec

The iExec platform [5] is the result of several research projects in the area of distributed systems at INRIA and CNRS. It collected over $12 million of investments in April 2017. The architecture of the platform is shown in Fig. 3.

The only difference from the standard service execution flow in the iExec platform is that after signing the smart contract, the consumer is required to divide the application into tasks and send them to the scheduler. I is not the scheduler that sends tasks to providers; the providers ask for new tasks when unoccupied.

A contract signed in Ethereum must contain its execution code and a description explaining its aim, a structure similar to requests for comments (RFC) defined by the Internet Engineering Task Force (IETF) and the Internet Society (ISOC). These contracts are then submitted for peer review and approved. Since smart contracts regulate the off-the-chain computation, an interface is needed to receive updates about the execution to correctly perform the actions defined in the smart contracts. The off-the-chain computation is coordinated by an open source desktop grid framework, namely XtremWeb-HEP [7], which covers data management, security, and deployment of resources using Docker, VMs on the provider's side, and updating this interface.

Recently, iExec introduced the concept of public and private worker pools. Providers can offer their resources in a public work pool, managed by a scheduler, and can also create private pools, where they rely on their own scheduler. A matchmaking algorithm (under construction) lists the work pools, and consumers can choose among them.

The proof-of-contribution module guarantees that the results of the computation are correct and is instrumental to improve consumers' trust; it plays the same role as the verification module of Golem.

IExec proposes an adaptive solution based on workers' reputation, replication of services, and financial incentives.[6] Consumers define the confidence required for the results, which is directly related to costs, and providers give a security deposit (stake). Tasks are duplicated and executed by randomly designed nodes until the confidence level exceeds the threshold defined by the consumer. The confidence level of the result is a function of the reputation and of the deposit of the nodes that execute the same task with the same result. At the end, the providers whose results have been verified split payments and stakes, and the reputation of providers with malicious behavior is decreased. The protocol is resilient against Sybil-like attacks [8] (attackers forge multiple identities to influence the system) because a stake from the nodes is required. However, no documentation is available about costs and overhead needed to meet the confidence level set by users.

iExec plans to focus on fog computing, and on supporting graphics processing unit (GPU) and memory guards for secure execution using the Intel SGX implementation. The project also provides an *Application Store*, where providers advertise their applications and consumers choose among different payment schemes, and a *Market Management Framework* is used to register bids and to provide templates. iExec supports the OpenCL framework to execute programs across heterogeneous hardware, and offers an application programming interface (API) for integration with other solutions. The creation of a data marketplace that certifies data exchanges is also planned.

SONM

SONM [6] is the most recent project among the three, and its ICO raised over $40 million in June 2017. Many aspects of the architecture and aims described in the original white paper have been reconsidered.[7] The project now focuses on IaaS, and it is planned to support PaaS. It is claimed

[6] Proof of contribution (PoCo) Analysis series: https://medium.com/iex-ec/poco-series-3-poco-protocole-update-a2c8f8f30126

[7] The team summarizes the changes here: https://blog.sonm.io/sonm-updates-and-specifies-its-roadmap-for-q4-2017-q2-2018-14d355cea6f

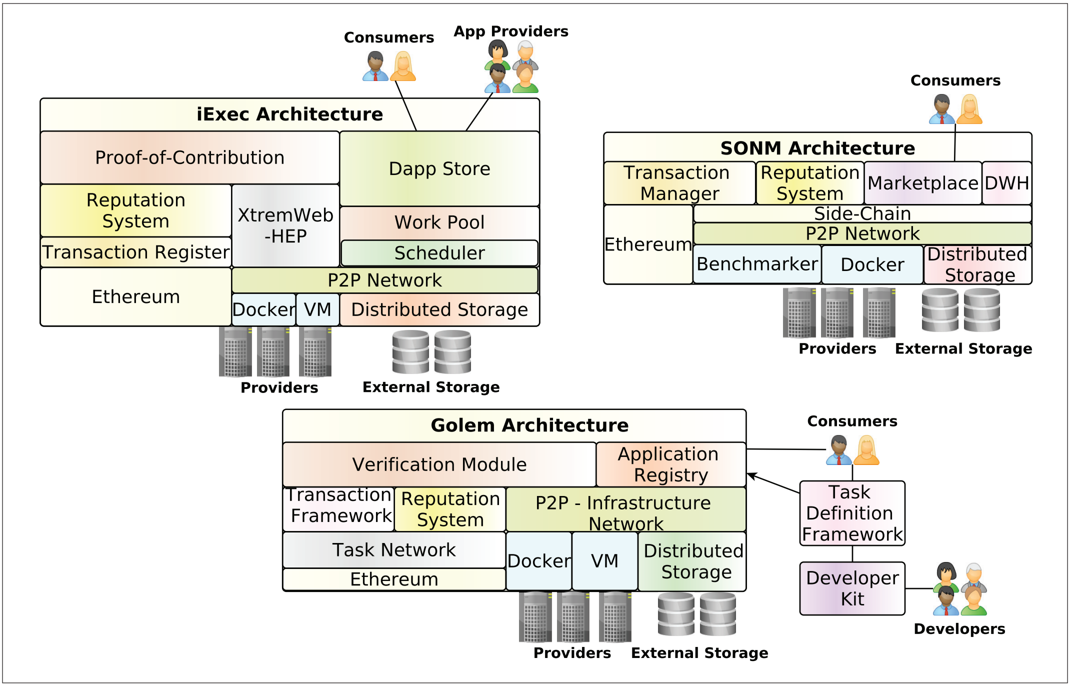

FIGURE 3. General architectures of Golem, iExec, and SONM.

Considering their architecture, the underlying technology, and the potential providers of the platform (which are geographically distributed and have relatively low computational power), we believe that it would be more appropriate to consider the three projects as decentralized fog computing solutions relying on different service layers.

that SONM could be used to provide infrastructure for the platform and the application layer of other projects like iExec and Golem. The architecture of SONM is illustrated in Fig. 3.

Different from the other projects, SONM adopts the transaction network only for payments and uses a side-chain (a clone of Ethereum) to cover functions, such as reputation, marketplace, and monitoring tasks. The marketplace matches service offers and requests to later be approved by consumers and providers. SONM will use Docker and Kurbenetes to manage containers. VMs will be supported, but no detail is provided. They also offer a simple set of benchmarks, which is executed when a provider enters the network. The data warehouse (DWH) component is a cache for data objects (e.g., profiles and orders) in the side-chain. Verification of service provision is performed only by the user, but there are plans to provide platform-level verification.

The main difference of the execution flow of SONM with the general schema is that before requesting a service, users transfer the funds in the transaction network to an SONM component called a gate, which communicates with the gate component of the side-chain that instantiates the same funds to be used in the side-chain, thus reducing transaction costs, which are relatively high in Ethereum.

Analysis and Challenges

The three projects have different visions: Golem is aimed at a decentralized supercomputer, iExec at a decentralized cloud, and SONM at a decentralized fog supercomputer. However, considering their architecture, the underlying technology, and the potential providers of the platform (which are geographically distributed and have relatively low computational power), we believe that it would be more appropriate to consider the three projects as decentralized fog computing solutions relying on different service layers.

Table 1 compares the projects and their underlying technologies.[8] Below, we expound what we consider important challenges for any project in the area and discuss incompatibilities between these projects.

•Ethereum, an implementation of blockchain technology, is currently an enabling standard for all three projects. It provides solutions for guaranteeing distributed consensus, defining smart contracts, managing payments, and guaranteeing identity. It alleviates the projects from the burden of developing specific solutions for these challenges and facilitates the integration with similar projects. The biggest drawback is the transaction costs. To reduce them, SONM uses Ethereum only to transfer funds to their side network. Golem, instead, uses a probabilistic scheme, but, although effective, it has strong requirements (a considerable number of consumers must adopt this scheme) and guarantees fair income distribution only in the long term and to providers that provide services frequently. Another challenge posed by Ethereum is scalability: it is currently restricted to 15 transactions per second, and this, in the long run, can impact on the considered projects. iExec plans to support other transaction networks and also to develop an independent one.

•Services are executed off the chain in a separate P2P network due to the computation cost in Ethereum. In the side-chain, the three projects use different but conceptually similar underlying technologies.

•Probably the biggest challenge for decentralized cloud solutions is *verification* of the computation, which is needed to avoid malicious actions by providers and consumers. Golem adopts three methods to check computation results: log verification, correctness checking, and redundant computation. Nonetheless, logs can easily be replaced; correctness checking works only in specific cases

[8] The table reflects the documentation of the project,s but since they are ongoing, they might still address the missing features or change underlying technology. Notably, the number of features does not reflect the project quality.

	Golem	iExec	SONM
Billing model	Pay-per-task	Pay-per-task	Pay-per-usage (time)
Cloud model	SaaS	IaaS, PaaS, SaaS	IaaS, PaaS
Communication	Whisper	P2P	Whisper
Computation platform	Own solution	XtremWeb-HEP	Own solution/Kubernetes
Data transfer	DAT	URI	BtSync
GPU support	✓	✓	✓
Memory guard	✓	✓	×
Open source	✓	✓	✓
QoS definition	×	✓	×
Reputation system	Own solution	Own solution	Own solution
Sandboxing	Docker, VMs	Docker, VMs	Docker, VMs
Service composition	×	×	×
Smart contract QoS	×	×	×
Transaction network	Ethereum	Ethereum	Ethereum/side chain
Verification	Log, correctness, redundant/high stakes	Redundant/high stakes	Planned

Table 1. Comparison of the blockchain-based decentralized cloud solutions.

and on some classes of problems, which could be detected by providers and used to deceive consumers; and redundant work, which requires extra computation, offers only a probabilistic guarantee, and works only for stateless applications. For example, in the machine learning stack provided by Golem as a use case, the whole computation has to be rerun to verify results. Moreover, some results, like those concerned with random numbers or random inputs, cannot be verified. iExec is planning to use redundant computations that have the drawbacks mentioned previously, which they want to mitigate using an escrow account with high stakes to penalize malicious users. Although SONM's white paper mentions the use of [9, 10] for verifying computation results, they have changed their focus to IaaS and currently do not provide solutions to verify if resources are actually provided to the consumer. We analyzed the mentioned solutions and noted several drawbacks. The solution of [9] requires redundant computing, the application code to be open, and the ability to stop and resume execution. The solution in [10] uses interactive protocols to probabilistically check proofs, but it is limited to a single class of computations and is costly for providers. Overall, these solutions only mitigate the computation verification problem, but its solution remains one of the main challenges for decentralized clouds.

•An important component of verification solutions is *reputation* or economic penalties for malicious behaviors. However, there must be a balance between the weight of reputations and the market entry cost. If reputation is too important, it becomes difficult for small and medium providers to enter the market. If reputation is ignored, attacks using forged identities become easier.

•Golem and iExec rely on oracles, which use the verification protocol described above to define whether the computation was successful and to trigger smart contracts to execute when terms are met. However, the oracle itself must be trusted since it can arbitrarily provide any information, irrespective of the truth. Therefore, the oracles used by the projects can directly influence the decision making process and reduce trust. There are several proposals for decentralized oracle[9] that could mitigate the dependence on central authorities.

•Quality of service (QoS) definition is essential for consumers, in particular for business consumers that require guarantees when outsourcing processes and for the automation of the service life cycle [11]. However, iExec is the only project that mentions QoS, but it uses standard service level agreements and does not consider dynamic aspects [12]. Moreover, according to the architecture of the projects, probably QoS will not be part of the smart contracts (*smart contract QoS*) and will be controlled only in the side-chain; no detail is provided on how it will be defined, monitored, and enforced.

•In decentralized environments, where any user can become a provider and resources may be very heterogeneous, comparing hardware and network performance is essential. iExec proposes a simple benchmarking approach at the application level, where sample applications of different categories are tested in a reference hardware. Then, for each request, iExec classifies the applications and defines their cost in terms of predefined categories and estimated execution time. More flexible approaches, for example, using machine learning techniques such as [13], can be adapted in this context. For SONM, simple IaaS benchmarking is proposed when providers enter the network, but no detail is provided. However, measuring the performance of providers when they enter the network might enable providers to improve their profile toward benchmarkers by cheating. For example, they could allocate better VMs and higher bandwidth when they know that they are interacting with benchmarkers and not with consumers.

•Consumers' data is managed by decentralized clouds and processed by third parties. Thus, solutions for guaranteeing *data privacy* are required, and regulations are becoming stricter. For example, the recent General Data Protection Regulation (GDPR) issued by the European Union could apply to the projects under consideration because they may be manipulating sensitive user data. However, the actual enforcement of GDPR might be hindered by the distributed nature of the systems and by the lack of central authorities. To comply with these regulations and improve data privacy, such projects need to provide some form of know your customer (KYC), to create mechanisms to guarantee the right to erasure, data anonymization, data protection in the design of business process, and data portability, and guarantee strict control on cross-border data transfer (e.g., only authorized toward countries with "adequate" legislation). Alternative solutions might need to be considered since blockchain and Ethereum are apparently incompatible with the right to erasure due to their data immutability property.

[9] http://www.oraclize.it/, https://www.smartcontract.com/

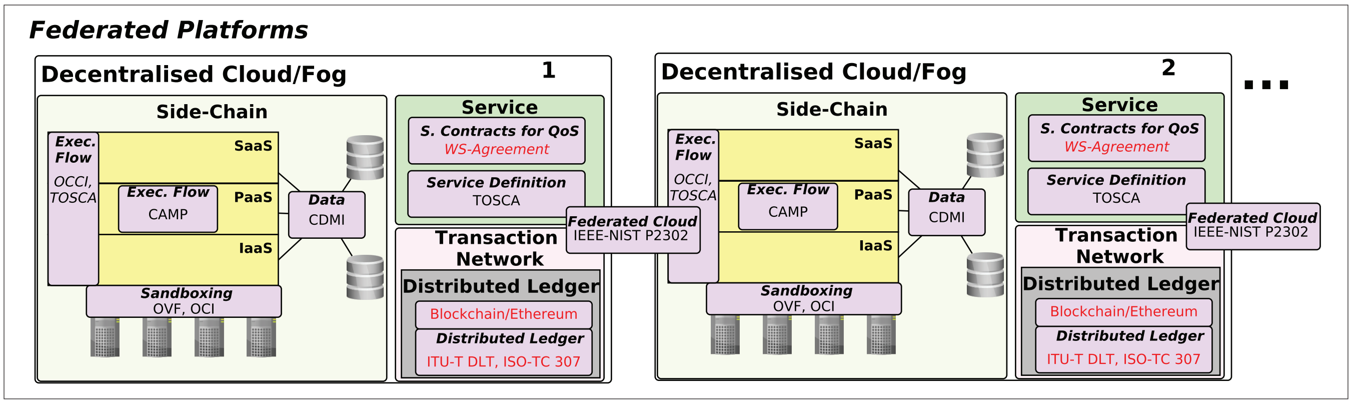

FIGURE 4. Standards for decentralized clouds.

STANDARDS

We believe that decentralized cloud projects could benefit from the presence of standards when dealing with the different aspects of service provisioning. In particular, we think that standards could be useful for defining QoS, automatizing service life cycle, facilitating portability of application, preventing data lock-in, complying with regulations, and creating open and integrated cloud markets.

The three analyzed projects are focused on market differentials rather than standards. Only iExec mentions collaborations with standardization bodies but only for smart contracts. However, the lack of standards gives rise to incompatibilities, which hinder integration. The main incompatibilities among the three projects arise from:

- *Service definition*. The projects do not provide details of services (e.g., how they get the necessary data or handle service dependencies), and each of them has different payment and pricing schemes.
- *Smart contracts for QoS*. None of the considered projects relies on smart contracts to specify QoS.
- *Execution workflows*. Golem and iExec require the selection of providers, while SONM assigns them automatically; moreover, iExec consumers have to define how to split services into tasks and execute them, while in Golem this information is provided by the system.
- *Management of components*. Currently, none of the projects considers *data structures*, *privacy*, *federation*, and *benchmarking*, or specifies the formats accepted by their *sandboxing* environments.
- *Identity and reputation*. Different methods of reputation assignment are used, and this, together with the absence of identity portability, limits reputation portability.

Now, we briefly describe the standards that could mitigate compatibility problems and discuss how they could help in facing the challenges defined in the previous section. Figure 4 situates the considered standards in the corresponding cloud components. The figure contains only areas (rectangles with purple background) for which there are ongoing standards initiatives. The standards in red partially cover the requirements of decentralized clouds.

Service definition. The Topology and Orchestration Specification for Cloud Applications (TOSCA) standard provides a framework that covers the structural aspects related to service templates and dependencies.

The considered projects provide marketplaces (app stores) for applications that can be executed in the project's cloud. However, we are not aware of standards that cover the description, pricing models, and other business aspects of these applications.

Smart contracts for QoS. When it comes to QoS guarantee as part of smart contracts, a de facto standard in clouds is the Web Services Agreement Specification (WS-Agreement [14]), which does not cover the dynamic aspects of services. We are not aware of works in this direction apart from our own [15].

Execution flow. TOSCA also specifies the operational aspects concerning the cloud stack (e.g., how and where to deploy the service). Open Cloud Computing Interface (OCCI), instead, provides a protocol and management API to define cloud management tasks, such as scaling and monitoring.

While TOSCA focuses on reusability, compositionality, and topology, OCCI focuses on the specification of runtime behaviours.

The OASIS standard Cloud Application Management for Platforms (CAMP) is a protocol for the development, deployment, and administration of application platforms, which simplifies moving applications between PaaS providers.

Management of components. The main sandboxing standards at the IaaS level are Open Virtualization Format (OVF), which covers definition, packing, and distribution of software to be executed in a VM, and Open Container Initiative (OCI), which aims at introducing standards for container format and runtime specification.

Data elements. The Cloud Data Management Interface (CDMI) standard defines interfaces for the management of cloud storage, definition of access control, and the creation, retrieval, and update of data elements; it can be exploited for distributed storage.

Data privacy. Regulations about the handling of user data, like EU's GDPR, have a direct impact on decentralized cloud. Considering them requires a joint effort of standardization and regulatory bodies. Fixing the data privacy standard would improve compatibility between the consid-

We believe that decentralized cloud projects could benefit from the presence of standards when dealing with the different aspects of service provisioning. In particular, we think that standards could be useful for defining QoS, automatizing service life cycle, facilitating portability of application, preventing data lock-in, complying with regulations, and creating open and integrated cloud markets.

ered projects and enable them to operate (e.g., in the European market).

Benchmarking. Benchmarking has only been partially addressed in the projects. Currently, there are no widely accepted standards. In decentralized clouds, there is an urgent need for a low computational cost benchmark to uniformly assess computing capacity of providers.

Federated clouds. To facilitate migration and pave the way to interoperability between providers, IEEE-NIST P2302 considers integration of providers/projects. It defines norms for protocols, business aspects, data exchange, and accounting, which could be used to integrate projects and create larger markets.

Distributed ledger. The term blockchain has become a synonym of distributed ledger, even if technically, blockchain is not the only technology for implementing them. For instance, Hashgraph and Tangle[10] are nonlinear solutions based on directed acyclic graphs. However, blockchain is currently the de facto standard, and almost all projects related to distributed ledgers (like the three analyzed here) use it. In fact, this commonality, supported by standards, can help when facing different challenges, like those concerned with *identity and reputation*. It can help by providing guarantees on the identity of the signatory party and by enabling the integration with projects for reputation management (like Synereo[11]) and projects for the definition of unique identities (like uPort and ERC725[12]). Several initiatives for the standardization of blockchain and distributed ledger in general are ongoing. The International Telecommunication Union — Telecommunication Standardization Sector Focus Group on Application of distributed ledgers and ISO/TC 307 is working on the definition of security, identity, interoperability, smart contracts, and architectural standards in the area. IEEE has promoted a major initiative[13] to disseminate standards in this area. However, all these initiatives are relatively new, and no concrete standard has emerged yet.

Conclusions

In this article, we have described the architecture and the implementation choices of three decentralized cloud systems. Moreover, we have compared them, discussed open research gaps in the area, and underlined the need for standardization of a number of features. Blockchain-based cloud projects are still in their infancy and need to address many open gaps, such as validation of the results of computations and benchmarking of providers' resources. The considered projects focus on the development of a functional product aimed at commercial solutions, and usability is one of their major success measures. However, in the long run, the lack of standards might become a barrier to competition with the large providers and might hinder the creation of open markets.

[10] https://www.hederahashgraph.com/, https://www.iota.org/

[11] https://www.synereo.com/

[12] http://uport.me/, https://github.com/ethereum/EIPs/issues/725

[13] http://blockchain.ieee.org

References

[1] S. R. Group *et al.*, "Big Four Still Dominate in q1 as Cloud Market Growth Exceeds 50%. Synergy Research Group," 2016.

[2] N. Szabo, "Smart Contracts: Building Blocks for Digital Markets," *EXTROPY: The Journal of Transhumanist Thought*, 1996.

[3] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System"; http://bitcoin.org/bitcoin.pdf, 2008.

[4] "The Golem Project Crowdfunding Whitepaper," The Golem Project, 2016, https://golem.network/doc/Golemwhitepaper.pdf

[5] "iExec Blockchain-Based Decentralized Cloud Computing v3.0," White Paper, iExec, 2018; http://iex.ec/whitepaper/iExec-WPv3.0-English.pdf.

[6] "SONM Supercomputer Organized by Network Mining," White Paper, SONM, 2017; https://github.com/masonicGIT/ico-whitepapers/blob/master/sonm/sonm.pdf.

[7] C. Cérin and G. Fedak, *Desktop Grid Computing*, CRC Press, 2012.

[8] J. R. Douceur, "The Sybil Attack," *Int'l. Wksp. Peer-to-Peer Systems*. Springer, 2002, pp. 251–60.

[9] R. Canetti, B. Riva, and G. N. Rothblum, "Practical Delegation of Computation Using Multiple Servers," *Proc. 18th ACM Conf. Comp. and Commun. Security*, 2011, pp. 445–54.

[10] S. T. Setty *et al.*, "Making Argument Systems for Outsourced Computation Practical (Sometimes)," *Proc. NDSS*, vol. 1, no. 9, 2012, p. 17.

[11] V. Scoca, R. B. Uriarte, and R. De Nicola, "Smart Contract Negotiation in Cloud Computing," *Proc. 10th IEEE Cloud Computing*, 2017, pp. 592–99.

[12] R. B. Uriarte, F. Tiezzi, and R. D. Nicola, "SLAC: A Formal Service-Level-Agreement Language for Cloud Computing," *Proc. 7th IEEE/ACM UCC*, 2014, pp. 419–26.

[13] R. B. Uriarte, S. Tsaftaris, and F. Tiezzi, "Supporting Autonomic Management of Clouds: Service Clustering with Random Forest," *IEEE Trans. Network and Service Management*, 2016.

[14] A. Andrieux *et al.*, "Web Services Agreement Specification (Ws-Agreement)," *Open Grid Forum*, vol. 128, no. 1, 2007, p. 216.

[15] R. B. Uriarte, F. Tiezzi, and R. De Nicola, *Dynamic SLAs for Clouds*, Springer, 2016; http: //dx.doi.org/10.1007/978-3-319-44482-6_3, pp. 34–49.

Biographies

Rafael Brundo Uriarte (rafael.uriarte@gmail.com) is a postdoctoral research fellow at IMT Institute for Advanced Studies, Lucca, Italy. His research interests include blockchain-based systems, cloud computing, autonomic computing, machine learning, and distributed systems. He has published in several international conferences and journals, such as *IEEE Communications Magazine*, IEEE/ACM CCGrid, IEEE CLOUD, and UCC. Additional information is available at http://rafaeluriarte.com/.

Rocco De Nicola is a professor of computer science at IMT School for Advanced Studies. His research is concerned with the foundations of distributed computing, the formal specification and checking of qualitative and quantitative properties of systems, and the protection of distributed systems and computer networks. He is the author of around 200 publications in international journals and books, and has been a keynote speaker at many international conferences and schools.