Polymorphic network structures allowing distributed clouds

Miloš Simić[0000-0001-8646-1569], Milan Stojkov[0000-0002-0602-0606], Tamara Ranković[0000-0002-5265-4626], Goran Sladić[0000-0002-0691-7392] and Miroslav Zarić[0000-0003-4671-3834]

Faculty of Technical Sciences, University of Novi Sad, Serbia
milos.simic@uns.ac.rs

Abstract. In recent years, we have witnessed the attempts of offering cloud services closer to data and end users. Even though there already are platforms on the market managing infrastructure at the edge, there are still many challenges yet to be resolved. One example would be serving requests from an optimal location. The upcoming real-time applications will be bound to rely on local data processing because of its sheer volume and strict time constraints, but this doesn't completely exclude the cloud from the picture. It will still be the best suited option for resource-intensive workloads. In this paper, we present the distributed cloud model that enables dynamic formation of ad-hoc clouds. By abstracting infrastructure to the level of software, we manage to streamline the process of resource reallocation, so that even the highly fluctuating needs of nearby population can be met. We also discuss the idea of data transfer from one distributed cloud to another when user mobility is detected. This strategy aims to minimize distance of users from data at all times, but the particular difficulty we focus on is how such a process can be executed with respect to data privacy requirements.

Keywords: distributed systems, cloud computing, distributed clouds, big data, data centers, infrastructure as software.

1 Introduction

In the recent decade, we are witnessing a slow paradigm shift of offloading computation and storage away from centralized infrastructures, such as cloud computing, to the edge of the network [1]. These infrastructures are not meant to replace cloud computing, but to complement it. They allow preprocessing and, in some cases, complete offload of the computation and storage from the cloud, so that only the key information can be transferred to the cloud for further analysis [1, 2].

Internet of Things and augmented reality applications, as well as autonomous vehicles or decision-making systems in Smart Grid require latency below a few tens of milliseconds, which a centralized cloud model at its current state cannot meet [1]. These applications generate large amounts of data that often cannot be easily moved from their source to the cloud for storage and processing [3].

Cloud computing is designed to allow data access, computation, or other utilities over the network [4], with infrastructure being placed on locations optimal for building

massive data centers. Furthermore, these utilities can be used and released on-demand [5], providing cloud providers with the benefit of offering numerous diverse services to its customers [6]. Cloud computing offers its services elastically according to the pay-as-you-go model [4, 6, 7]. This model is preferred by many companies as it eliminates the need for costly up-front investments, required to build and maintain huge data centers [8], regardless of the vast distance from the nearest cloud data center. Because this approach can lead to potentially too-high latency in the system [9], we must have these implications in mind [2, 10].

Nevertheless, by modifying the current cloud model, we can aid it in dealing with challenges that are to come by extending the processing and storage capabilities of the cloud to the edge of the network. With this idea, we are able to process data at its source [11] and transfer only valuable information to the cloud for sophisticated analysis and decision-making, where significantly more resources are available [12].

By moving clouds closer to the ground and data sources [11], we challenge the traditional cloud infrastructure of massive data centers with aggregate resources that could lower the administration cost of the entire system. Vogels et al. defined cloud computing as an aggregation of the two elements - computing resources as a utility and software as a service [6]. If we adhere to this definition, we can conclude that we are not moving away from the original idea of cloud computing. On the contrary, we are merely pushing it one step further, allowing the creation of cloud-like structures closer to the users [2, 10].

Moving clouds closer to the ground opens new research areas such as ad hoc cloud-like computing infrastructure creation and repurposing of resources to best meet the needs of the local population. Another idea worth considering is that users become no longer locked into offerings of a single vendor. They do not need to rely solely on the centralized infrastructure of traditional clouds. This idea opened the door for distributing clouds closer to the ground, creating a distributed clouds model. On the other hand, users are familiar with applications developed for the traditional cloud infrastructure – cloud-native applications. Distributed clouds need to offer an application model that is intuitive and familiar to the users. These applications must be able to migrate between traditional and distributed clouds as needed.

Another motivation factor for developing the distributed cloud model are scenarios of governments forbidding sensitive information to leave country borders in order to prevent misuse [13, 14]. Protection of data privacy must be done responsibly, from collection to storage. In developed countries, once the data is collected, different laws and regulations start to apply depending on the context of the collection. Sometimes, cloud computing cannot answer this problem efficiently or adequately. This problem is due to cloud computing infrastructure and how it is built. In some cases, the nearest data center is outside the country's jurisdiction forcing sensitive information to go beyond borders [14]. This is another challenge the new infrastructure model needs to address.

Contributions of this paper can be summarized in the following way:

- We present an idea of complementing traditional cloud infrastructure by processing data closer to the data sources and transferring only valuable information to the cloud
- We claim that it is possible to create ad hoc cloud-like infrastructure as disposable elements that can be repurposed as needed to serve local population needs
- We introduce the idea for users to use this infrastructure similarly as they use the traditional cloud and move their application seamlessly from the traditional cloud to distributed clouds and vice versa
- We address the privacy challenges that arise with data in the new infrastructure proposal

The rest of the paper is structured as follows. Section 2 describes related research on the topic of distributed clouds. Section 3 presents the idea of distributed clouds at the edge of the network. In Section 4, different aspects of the creation and maintenance of the model are given. Section 5 presents data privacy restrictions the model needs to address to become a feasible solution. In Section 6, we conclude the work done

2 Related work

In the era of cloud computing and service-oriented architectures, automation is one of the key elements. Various tools are developed, allowing abstraction infrastructure to the level of software. Generally speaking, there are two available options [15] on how information and actions could be sent to the system: (1) imperative and (2) declarative.

Imperative tools usually rely on some language where users specify how, and in some cases, in which order actions need to be executed onto the system. The negative side of this approach is that this strategy is more error-prone. Users may introduce a bug in the system that might be hard to debug and find. On the other side, the positive thing is that these tools have existed for a long time, and best practices are more precise and readily available to the users. Typical representatives of such tools are Chef, Ansible, and Puppet. On the other hand, declarative tools shine in much more complicated environments such as cloud computing. These tools usually rely on independent platform and language configurations without defining explicit actions. The users specify the desired state and submit it to the system. And then, the system determines the optimal way to achieve it. These tools usually rely on immutable infrastructure to achieve defined goals [16, 17]. Typical representatives of such tools are Terraform, Polumni, or CFEngine.

Existing tools lack support for distributed clouds. Although both of them could potentially be extended to support new environments, in this research, we

will present the possibility of a native tool based on declarative ideas and the creation and repurposing of pools of resources in the form of distributed clouds.

Besides the tools mentioned earlier, in recent years new paradigm emerged called Osmotic computing. This paradigm proposes application decomposition into microservices for one reason to exploit resources in both edge and cloud infrastructures [18]. Osmotic computing applications should be developed in such a way as to equalize the microservices concentrations on the two sides. The proposed model could fit well into the osmotic computing model, allowing dynamic and efficient management of distributed cloud infrastructure.

In their research, Greenberg et al. [19] introduced the exciting concept of Micro data centers. The authors defined them as data centers that operate nearby serving only the population nearby, and as such, they can minimize the latency and costs for end-users [19, 20] on the one side, while at the same time they reduce fixed costs that traditional big data centers have. The minimum size of these data centers is defined strictly with the needs of the local population [19, 21]. The key feature of these data centers is the agility to dynamically grow and shrink resources as needed while satisfying the demands and resource usage from the optimal location [19].

Ciobanu et al. [22] introduce drop computing, an interesting idea able to compose edge computing platforms ad hoc, thus enabling collaborative computing dynamically. Drop computing employs the mobile crowd formed of nearby devices, and instead of sending requests to the cloud, it routes them to the mobile crowd, enabling quick and efficient access to resources. The drop computing model may raise a few concerns, but it can dynamically include crowd nodes when extra storage or processing power is needed.

The practical applicability of proposed solutions might be in question when data collected, processed, and stored contains Personally Identifiable Information (PII). If PII needs to be stored on different edge locations to utilize computing on the edge, security and privacy issues must be addressed [23]. Takabi et al. in [24] describe numerous challenges in traditional cloud computing that apply to distributed clouds in terms of data protection used in heterogeneous shared infrastructures. To address these challenges, especially privacy-related ones, a formal introduction was made through different legislation requirements, most notably the General Data Protection Regulation (GDPR) for European Union countries, to increase the safety of sensitive personal data of EU citizens. This was done by restricting the flow of sensitive data outside of the EU [25]. GDPR led to the development of a framework with a set of policies and rules that can be applied to the cloud and edge technologies stacks called GAIA-X [26].

The model proposed in this paper differs from similar solutions as it allows dynamic formation of fully-disposable distributed clouds. Existing tools, such as Kubernetes and Mesos, are designed to handle failures inside a cluster, deeming no server or cluster irreplaceable. We go one step further, treating the entire clouds completely replaceable. This approach opens numerous possibilities for optimizing the infrastructure by strategically constructing multiple clouds designed for failure. We hide the complexities of

infrastructure management behind the Infrastructure-as-Software interface users interact with. This contributes to the usability of our solution and enables clients to rely on well-established principles, including reusability, modelling, testing, and evaluation.

3 Distributed clouds model

In this section, we will explain the high overview of the model and the idea of distributed clouds at the edge of the network, influenced by architectural ideas from traditional clouds

3.1 Distributed clouds infrastructure model

We can accept the definition of cloud computing given by Vogels et al. as an aggregation of the two elements - computing resources as a utility and software as a service [6]. These resources are offered to clients as needed elastically. To achieve such elasticity, cloud infrastructure is organized into three building blocks: (1) cluster, (2) region, and (3) availability zones [27].

Once set, these building blocks are hard to rearrange because they are part of the physical infrastructure. That may result in some customers far away from the nearest data center that serves real-time applications having difficulty dealing with latency.

However, such organization of building blocks may be a good starting point for building distributed clouds model with some adaptations. The important thing to consider is that distributed clouds must operate over arbitrary vast geographic areas serving nearby populations. The user requests should first reach the distributed clouds and then, if necessary, route to the traditional cloud.

In traditional cloud computing, the region is a physical element of the existing infrastructure [27]. In distributed clouds, a region is a logical element composed of one or more physical clusters of nodes over an arbitrary vast geographic area. Like traditional cloud regions, distributed clouds regions should be capable of accepting and releasing clusters of machines if needed.

Greenber et al. [19] described a micro data center's optimal size as big as the population nearby requires. This description is implied in distributed clouds. As a result, we get that a single distributed clouds region can be composed of at least one cluster, but the cloud spans multiple clusters achieving higher availability, resilience, storage, and processing capacity.

On the highest level of abstraction, we have a topology. Topology can be viewed as a single cloud provider compared to traditional clouds. Topology is capable of having multiple regions but must have at least one. As regions, topology can also add or remove regions as needed. For all three abstractions, cluster, region, and topology, the vast distances should be avoided, at least in normal circumstances. The only physical thing in distributed clouds model is the node inside the cluster. All other, higher abstractions should be created descriptively.

Compared with micro data centers and nano data centers, with previous claims, we are getting that distributed cloud clusters can become either of the two and everything in between, thus, giving users a higher space for better optimization. On the other side, the distributed cloud model can act as the traditional cloud computing or edge computing model, but does not have to become either. Table 1 depicts the differences and similarities between traditional and distributed clouds in terms of the physical and/or logical elements in the two architectural patterns.

Traditional cloud computing is here to stay, and it can only benefit from extension by distributing processing and storage capabilities to the edge of the network. Therefore, traditional cloud, distributed clouds, and edge computing or clients are building three-layer cloud architecture.

Table 1. Infrastructure comparison between the clouds

Concept \ Cloud type	**Traditional**	**Distributed**
Logical	Cloud provider	Topology
		Region
Physical	Region	Cluster
	Zone	

It is important to notice that in this three-tier cloud infrastructure, we have some benefits and some drawbacks at every level. As we go up the levels, we have many more available resources to utilize, but on the other hand, response time is higher since big data centers might be far away from the client. As we go down the levels, we have fewer resources available for utilization, but the response time is faster, as depicted in Fig. 1.

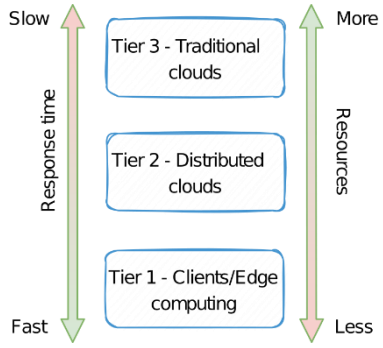

Fig. 1. – Three tier architecture and resource availability

Since distributed clouds operate in proximity to data sources and nearby users, they venture into geo-distributed systems. Furthermore, previously described abstractions clusters, regions, and topologies allow us to cover arbitrary

vast geographic areas, with the ability to shrink or expand these abstractions as needed, forming new pools of resources for applications to utilize.

So new microservice applications should utilize these architectural patterns as needed. Applications could be split into the front and back processing tiers, where front processing is in distributed clouds and closer to the clients, while the back processing tier utilizes traditional cloud infrastructure. These applications could be moved seamlessly between traditional and distributed clouds as needed, relying on osmotic computing ideas [18]. Front-tier applications can store and process data on the source and transfer only useful data to the traditional cloud, and the backend tier can be used for more complex and deeper analysis. The communication between these application parts is crucial for new-age real-time applications.

4 Policy based computing

This section will describe different aspects of forming and maintaining distributed clouds model. We will see in what circumstances we can dynamically create infrastructure in proximity to users, how we can handle data if users leave their current position, and how to automatically scale infrastructure, all based on user-defined policies, entering policy-based computing.

4.1 Infrastructure management

Cloud computing infrastructure is fixed and created in strategic locations worldwide to serve as many users as possible. If we want to expand infrastructure with new resources, this requires connecting modules physically to the rest of the infrastructure [28]. On the other hand, distributed clouds do not have that rigid infrastructure. Distributed clouds require dynamic interactions between the elements of the infrastructure. However, also, users should be able to create and repurpose infrastructure dynamically.

In order to achieve such elasticity, we need to abstract infrastructure to the level of software and venture into infrastructure programming [29]. Such an approach yields reusability of the available tools, principles, and techniques (e.g., testing, modeling, and evaluation) [30].

Existing orchestration tools such as Kubernetes, Apache Mesos, and Docker Swarm work at the cluster level "treating servers as cattle, not pets" (i.e., numerous servers built using automated tools designed for failure, where no server is irreplaceable) [31]. One cluster can spread to multiple zones, reducing the chance that a failure in one zone disrupts services in other zones. Kubernetes even allows multi-cluster, treating clusters as disposable elements. The model presented in this research goes a step further, proposing the creation of distributed clouds computing model for single use as disposable elements. It gives users more dimensions for the operation and optimization of infrastructure and services.

We must be aware that the deployment of such infrastructure will not happen fast. The key to success is to simplify infrastructure management [32]. Manually setting up the infrastructure is not acceptable, especially in a geo-distributed environment. To achieve such a level of freedom, in infrastructure creation, we must create at least four protocols [2, 10]: (1) health-check protocol that gives the whole system state of every registered node, (2) cluster formation protocol that forms new clusters, and (3) list detail protocol that shows the current state of the system to the user, and (4) query nodes protocol that will show free nodes to the user based on some criteria.

The one important aspect of these protocols that we cannot overlook since they will be implemented in such a complex environment is that they need to be formally correct. They need to be modeled so that, by design, they do not leave room for potential errors.

Infrastructure creation and repurposing should be done descriptively. The users describe their desired infrastructure using some of the existing file formats, submit it to the system, and let the system deal with the rest.

4.2 Data handling

Distributed clouds rely on the data locality principle – moving the computation closer to the data instead of moving data to the computation [33]. With this idea, distributed clouds can serve user requests from the best location – the closest distributed clouds to the user.

Equation (1) states that distributed clouds clusters could be as wide as the whole city or as small as all devices in a single household and everything in between [2]. Furthermore, we need to be careful and aware that we might not have the resources to store all the data, or the users may leave their place, city, or state, for example, and they should be able to access personal data. That is one of the strong points of the traditional clouds, and as such, we want to preserve this feature.

The equation (2) asks what options we have for data retention or how long data can stay in the system close to the users. If we go more towards the edge computing side, we have less storage capacity, meaning that only the most recent data should be stored nearby users while other older data should be stored in the traditional cloud because of the available resources. Together, equations (1) and (2) give us two functions to optimize data retention in the system and how much of the data we will store in distributed clouds.

The second aspect we need to investigate is what will happen to users' data if they move far away from the clouds nearby. That should be part of the agreement between distributed clouds providers and the users, and it could be specified in some policies, e.g., how much of the data should follow users and does data follow the users in the first place.

If we want to transfer data from one region to another, we could use traditional clouds as the backbone. If users allow, data will be transferred from

distributed clouds to the traditional cloud. Furthermore, user data can be transferred to other places using previous equations and defined policies.

With data transfer from place to place, we have to be careful. We cannot transfer all the data from place to place because that will potentially introduce high latency in the system. So only the newest data should be transferred at once, and then as a user requires more data, we can pull what is needed from the traditional cloud and cache it to the distributed clouds for future use. Data transfer should be part of distributed clouds policies and could be offered at different levels to the users.

4.3 Infrastructure creation and autoscale

All nodes will inform the system about their state and health via the healtcheck protocol. Free nodes, which are not used in other clusters, regions, or topologies, will be listed to the user. Users can query them using a system of labels. Labels are sets of key-value pairs in free text form, and labels describe nodes for the users. For example, labels maybe contain information about resources, operating systems, location, or architecture.

In order to create new infrastructure (clusters, regions, and/or topologies), the user first needs to create a query with specific criteria. These submitted criteria are checked against the labels of every node. Only free nodes with all specified labels will be part of the result returned to the users. These nodes will be reserved for some time TR, giving the user time to start the mutation protocol and create infrastructure. Fig. 2 shows query protocol communication.

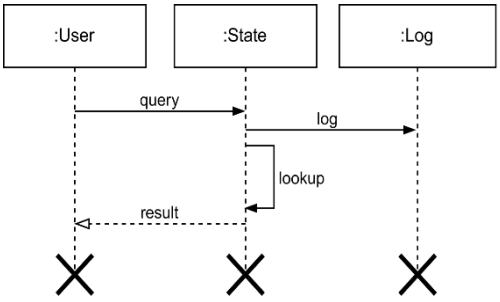

Fig. 2. Query protocol.

If a node desire to be part of the system pool of resources, it needs to obey a few rules:

- a node must run an operating system with a usable file system;
- a node must be able to run some isolation engine for applications (e.g., containers, virtual machines, or unikernels);
- a node must have available resources for utilization for processing, applications, or data storage;
- a node must have an internet connection;

- a node must provide a list of attributes in the form of a list of key-value pairs – labels.

Even though these rules look too simple at first glance, they are well-defined and chosen with purpose. When we exhaust all existing resources and have a sudden urge for new resources (e.g., to support bursts of requests or catastrophic events), the model can be extended, allowing the inclusion of volunteer nodes to depreciate load for an indefinite period. When these resources are not needed anymore, our model can release them back to a pool of free resources, similar to the cloud model.

The user should define a new system state using the description file. When the new state is accepted and ready to be processed, the system will start infrastructure creation protocol (Fig. 3).

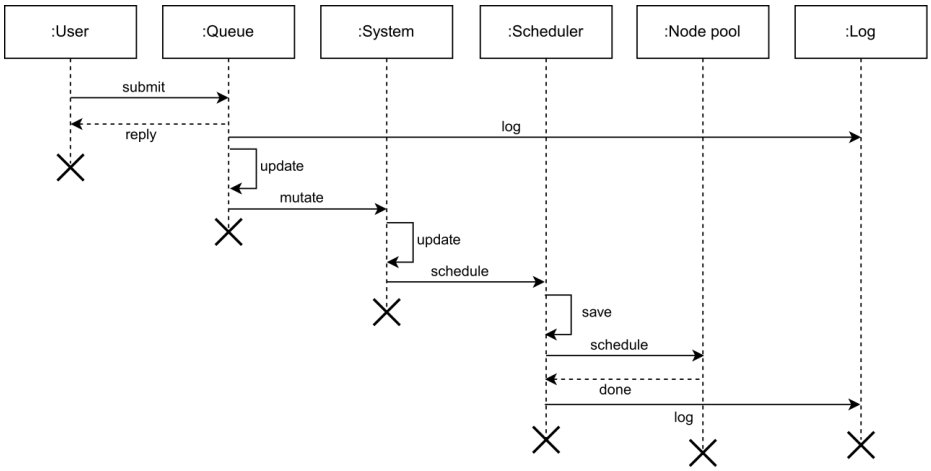

Fig. 3. Infrastructure creation protocol.

Once the infrastructure is created, the creation process is considered done. When the user submits a new state to the system, this task will be accepted and registered with the PENDING state. If an error occurs (e.g., no available resources or nodes), the task will change to FAILED. If there are no errors, the system will start processing the task, and the state will be IN PROGRESS. When the infrastructure creation is finished without errors, the task state will be CREATED, otherwise FAILED. Fig. 4 shows task state diagram.

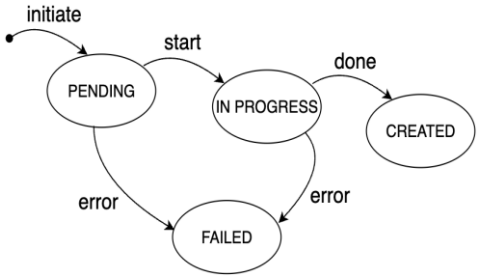

Fig. 4. Task state diagram.

The system will continue to monitor the actual state and to compare it with the desired state using reconcile pattern [34]. The reconcile pattern contains a loop that will constantly compare desired and actual states, and if the two differ, it will try to reconcile them. The node state is provided using the health-check protocol (Fig. 5), so if the system does not hear from some node for some extended period, we can consider that this node is down, and we need to act upon it.

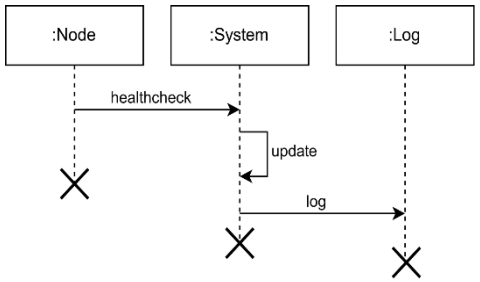

Fig. 5. Health-check protocol.

That opens an interesting area for automatic infrastructure scaling in two cases: (1) node(s) failure and (2) increased/decreased traffic. In order to fulfil these options, distributed clouds providers can create different policies for both options, and users can choose them based on their needs.
In the first case, we can apply the find-and-replace strategy. Here system tries to find similar node(s) based on user-defined criteria and tries to add them to already created infrastructure. If the failed nodes return online after some time, the system can release unnecessary nodes. In the second option, the system can decide that infrastructure needs more or fewer nodes, based on demand, and automatically adjust this by adding new nodes or removing existing ones when needed. This policy needs to be explicitly added by the user because it may increase the overall cost of the system. That is especially the case with adding new nodes to meet new demands. Furthermore, the reconcile loop may be beneficial in both cases because we are getting fully automated infrastructure without much-needed user interaction.

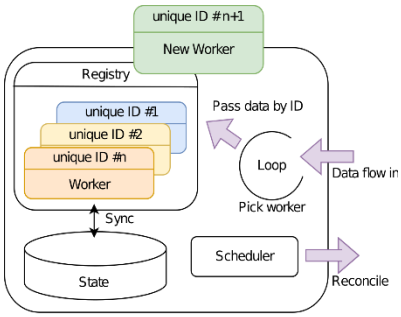

Fig. 6. Reconcile loop and controller extension.

The same pattern may be used for infrastructure and applications as well. Moreover, extending such a system is easy because we only need to add a new controller to the reconcile loop. Fig. 6 shows reconcile loop with controller extension.

5 Privacy by design

In the Internet Security Glossary, data privacy is described as the right of an entity (normally a person), acting on its behalf, to determine the degree to which it will interact with its environment, including the degree to which the entity is willing to share information about itself with others [35]. Information can take many forms, from user-submitted data through web forms to financial transaction data. That is why it is essential to detect what information systems will use and make data classification accordingly. Data classification will identify data elements with respect to their value in the business. Having defined use cases for data usage is vital to satisfy the Privacy by design principle. The easiest way to avoid protecting sensitive data is to avoid collection altogether or to practice proportionality by collecting only necessary information to accomplish a defined goal. Shaikh et al. [36] defined three types of characteristics on which data has to be classified to be able to protect it and preserve privacy: (1) content, (2) access control, and (3) storage.

The data's content has multiple features, such as accuracy, consistency, validity, and completeness. These features can be used to make data classification adequately. There are two types of data localization: soft and hard [37]. Soft data localization involves" mirroring" that requires some data to be stored within some area but also allows that data to be copied and used elsewhere. Hard data localization enforces that critical sensitive data and PII must be stored within certain borders, and further transfer is not allowed. Access to the content can be managed with adequate access control and encryption.

Since distributed clouds rely on the data locality principle, access to the data must be geographically regulated. Various access control models are used in traditional infrastructures, primarily based on Role-Based Access Control (RBAC).

The spatially aware variant that can be used in our model proposal is GEO-RBAC [38]. GEO-RBAC spatial features allow entities to be mapped onto locations in the given space. In this way, the visibility and accessibility of data in a particular region can be controlled. That is ideal for scenarios where data should not reside outside country borders. In previous chapters, we mentioned data transfer as a real possibility when user movement is detected. Instead of relying on distribution between distributed and traditional clouds, we can limit the distribution to other distributed clouds regions that formally reside inside allowed borders. That can be done by extending the cluster formation protocol with MOVE operation that allows transferring workload and data to another similar idling cluster following the policies defined with the GEO-RBAC scheme to maintain user data availability. As shown in Fig. 7, the MOVE operation is the responsibility of the Workload shifter component (WS) that transfers data from cluster A (ClsA) to cluster B (ClsB), similarly as described in our previous work [14].

When it comes to storage, data-at-rest encryption ensures that data is not stored as plain text. Here, data written to disk is encrypted using a set of secret keys known only to privileged users of the system. That protects data from compromise if a malicious user gains access to the storage system. Any form

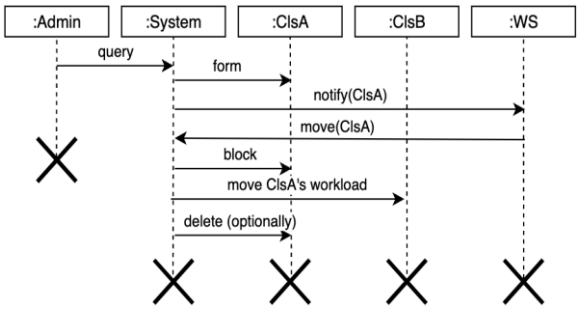

Fig. 7. Protocol extension with the Workload shifter component (WS).

of encryption with well-known industry-approved algorithms introduces performance penalties [39]. This form of protection can introduce a higher level of complexity because of the key management since keys need to be supplied by a user at system startup and restart. Also, keys can be stored within the provider, potentially meaning that their exact location might be outside of legally defined borders where the data resides. If there is an absence of policy that guarantees the protection and location of secret keys, an alternative should be provided. To overcome this shortcoming, bring-your-own-key (BYOK) can be enforced where the user or organization that uses distributed clouds owns the key and is solely responsible for its storage and security [40]. This way, users can revoke access to encryption keys without vendor reliance and forbid further personal information processing.

Another important feature that must be available when we talk about storage is a data backup and recovery plan. Based on the criticality of the data defined with data classification, a backup plan should be associated. The extension that the MOVE operation gives us can also be used for this purpose. Based on data classification, critical data can be replicated in adjacent clusters

6 Conclusion

This paper presents one solution for the dynamic creation of distributed cloud-like infrastructure that will serve requests from users nearby and cooperate with the traditional cloud to solve many challenges in new-age real-time applications. We used the idea of traditional cloud infrastructure but adapted it for a different environment, creating a familiar computing model for the users. The infrastructure is created dynamically using infrastructure as a software principle to abstract infrastructure to the level of software, making infrastructure programming easier to implement. This dynamic organization of nodes allows us to provide resources where they are needed and cover arbitrary vast geographic areas, with the ability to organize resources situationally.

The drawback of such a solution is that it might require substantial initial investments, but existing cloud providers or governments may offer it as a service to its users, as any other service in the cloud. As part of our further work, we plan to extend the proposed model with namespacing, allowing multi-tenancy in the system and remote configuration and control of the system

Acknowledgements. Funded by the European Union (TaRDIS, 101093006). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

1. M. Chiang and T. Zhang, “Fog and iot: An overview of research opportunities,” IEEE Internet Things J., vol. 3, no. 6, pp. 854–864, 2016.
2. M. Simić, I. Prokić, J. Dedeić, G. Sladić, and B. Milosavljević, “Towards edge computing as a service: Dynamic formation of the micro data-centers,” IEEE Access, vol. 9, pp. 114468–114484, 2021.
3. J. Cao, Q. Zhang, and W. Shi, Edge Computing: A Primer. Springer Briefs in Computer Science, Springer, 2018.
4. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the clouds: A berkeley view of cloud computing,” Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.
5. Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” J. Internet Serv. Appl., vol. 1, no. 1, pp. 7–18, 2010.

6. W. Vogels, “A head in the clouds the power of infrastructure as a service,” in Proceedings of the 1st Workshop on Cloud Computing and Applications, 2008.
7. M. Satyanarayanan, “The emergence of edge computing,” Computer, vol. 50, no. 1, pp. 30–39, 2017.
8. S. A. Monsalve, F. G. Carballeira, and A. Calderón, “A heterogeneous mobile cloud computing model for hybrid clouds,” Future Gener. Comput. Syst., vol. 87, pp. 651–666, 2018.
9. S. K. A. Hossain, M. A. Rahman, and M. A. Hossain, “Edge computing framework for enabling situation awareness in iot based smart city,” J. Parallel Distributed Comput., vol. 122, pp. 226–237, 2018.
10. M. Simić, G. Sladić, M. Zarić, and B. Markoski, “Infrastructure as software in micro clouds at the edge,” Sensors, vol. 21, no. 21, 2021.
11. A. Khune and S. Pasricha, “Mobile network-aware middleware framework for cloud offloading: Using reinforcement learning to make reward-based decisions in smartphone applications,” IEEE Consumer Electron. Mag., vol. 8, no. 1, pp. 42–48, 2019.
12. H. Ning, Y. Li, F. Shi, and L. T. Yang, “Heterogeneous edge computing open platforms and tools for internet of things,” Future Gener. Comput. Syst., vol. 106, pp. 67–76, 2020.
13. N. Cory, “Cross-border data flows: Where are the barriers, and what do they cost?,” 05 2017.
14. V. Indjić, M. Kovačević, M. Simić, and G. Sladic, “Towards local cloud infrastructure in developing countries as a response to data localization regulations,” 01 2022
15. M. Wurster, U. Breitenbücher, M. Falkenthal, C. Krieger, F. Leymann, K. Saatkamp, and J. Soldani, “The essential deployment metamodel: a systematic review of deployment automation technologies,” SICS Softw.-Intensive Cyber Phys. Syst., vol. 35, no. 1-2, pp. 63–75, 2020.
16. M. Perry, The Art of Immutable Architecture: Theory and Practice of Data Management in Distributed Systems. Apress, 2020.
17. P. Helland, “Immutability changes everything,” Commun. ACM, vol. 59, no. 1, pp
18. M. Villari, M. Fazio, S. Dustdar, O. Rana, D. N. Jha, and R. Ranjan, “Osmosis: The osmotic computing platform for microelements in the cloud, edge, and internet of things,” Computer, vol. 52, no. 8, pp. 14–26, 2019.
19. A. G. Greenberg, J. R. Hamilton, D. A. Maltz, and P. Patel, “The cost of a cloud: research problems in data center networks,” Comput. Commun. Rev., vol. 39, no. 1, pp. 68–73, 2009.
20. C. Shi, K. Habak, P. Pandurangan, M. H. Ammar, M. Naik, and E. W. Zegura, “COSMOS: computation offloading as a service for mobile devices,” in The Fifteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc’14, Philadelphia, PA, USA, August 11-14, 2014 (J. Wu, X. Cheng, X. Li, and S. Sarkar, eds.), pp. 287–296, ACM, 2014.
21. N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” IEEE Internet Things J., vol. 5, no. 1, pp. 450–465, 2018.
22. R. Ciobanu, C. Negru, F. Pop, C. Dobre, C. X. Mavromoustakis, and G. Mastorakis, “Drop computing: Ad-hoc dynamic collaborative computing,” Future Gener. Comput. Syst., vol. 92, pp. 889–899, 2019.

23. B. Varghese and R. Buyya, “Next generation cloud computing: New trends and research directions,” Future Generation Computer Systems, vol. 79, pp. 849–861, 2018.
24. H. Takabi, J. B. Joshi, and G.-J. Ahn, “Security and privacy challenges in cloud computing environments,” IEEE Security & Privacy, vol. 8, no. 6, pp. 24–31, 2010.
25. European Commission, “Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance),” 2016.
26. “Gaia-x.” https://gaia-x.eu, Accessed: 2024-01-14.
27. F. R. de Souza, C. C. Miers, A. Fiorese, M. D. de Assunção, and G. P. Koslovski, “Qvia-sdn: Towards qos-aware virtual infrastructure allocation on sdn-based clouds,” Journal of Grid Computing, vol. 17, pp. 447–472, Sep 2019.
28. J. R. Hamilton, “An architecture for modular data centers,” in CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings, pp. 306–313, www.cidrdb.org, 2007.
29. F. Brian, F. Nicole, S. Klaas-Jan, H. Jez, and D. Brian, “Infrastructure is software too!,” SSRN Electronic Journal, 01 2015.
30. M. Artac, T. Borovsak, E. D. Nitto, M. Guerriero, and D. A. Tamburri, “De-vops: introducing infrastructure-as-code,” in Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017 - Companion Volume (S. Uchitel, A. Orso, and M. P. Robillard, eds.), pp. 497–498, IEEE Computer Society, 2017.
31. P. Andrade, B. Tim, J. Eldik, M. Gavin, B. Panzer-Steindel, M. Santos, and U. Schwickerath, “Review of cern data centre infrastructure,” Journal of Physics Conference Series, vol. 396, pp. 2002–, 12 2012.
32. M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” IEEE Pervasive Comput., vol. 8, no. 4, pp. 14–23, 2009.
33. Z. Guo, G. C. Fox, and M. Zhou, “Investigation of data locality in mapreduce,” in 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012, Ottawa, Canada, May 13-16, 2012, pp. 419–426, IEEE Computer Society, 2012.
34. M. Luksa, Kubernetes in Action. Manning, 2018.
35. “Internet security glossary - privacy.” https://www.ietf.org/rfc/rfc2828.txt, Accessed: 2024-01-14.
36. R. Shaikh and M. Sasikumar, “Data classification for achieving security in cloud computing,” Procedia Computer Science, vol. 45, pp. 493–498, 2015. International Conference on Advanced Computing Technologies and Applications (ICACTA).
37. J. Daskal and J. Sherman, “Data nationalism on the rise,” Data Analyst, 2020.
38. M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca, “Geo-rbac: A spatially aware rbac,” ACM Trans. Inf. Syst. Secur., vol. 10, p. 2–es, feb 2007.
39. A. Gomes, C. Santos, C. Wanzeller, and P. Martins, “Database encryption for balance between performance and security,” 2021.
40. V. Kumar and I. Sharma, “Bring-your-own-encryption: How far are we?,” in 2016 11th International Conference on Industrial and Information Systems (ICIIS), pp. 672–677, 2016.