LightKone: Towards General Purpose Computations on the Edge

Ali Shoker*, Joao Leitao†, Peter Van Roy, and Christopher Meiklejohn‡

December 21, 2016

Abstract

Cloud Computing has become a prominent and successful computing model. However, the immense volumes of data generated through social networks, data analytics, and the Internet of Things impose new challenges that surpass the capability of Cloud Computing infrastructures. For example, the data generated in an IoT scenario can overwhelm a data center if one is not careful to do some computation locally. Additionally, this can also lead to unavailability scenarios when devices lack the connectivity to reach the data center. The Edge (a.k.a. Fog) Computing model extends the Cloud Computing model to exploit resources at the edge of the network (closer to the clients), avoiding the necessity of executing applications fully on the cloud. This approach yields additional benefits for applications including privacy, availability, and local decision making. In this work we present the foundations of a new Horizon 2020 research project called LightKone that extends Edge Computing for general-purpose computation, making it more scalable and flexible, and providing a set of mechanisms to simplify development and deployment of applications. The LightKone approach is based on the use of synchronisation-free shared mutable data combined with robust and efficient hybrid gossip communication primitives. In this way, we intend to build full systems that are completely synchronisation-free.

From Cloud to Fog/Edge Computing

With the notable growth of the Internet and web services, data centers with high computational and storage capacities became crucial and the primary architecture employed by technology giants and large-scale businesses to support their applications and operation. However, the complexity and cost associated with the maintenance and scalability of these architectures lead to the emergence of the Cloud Computing model that provides "everywhere, anytime" properties via scalable data centers built

* *HASLab, INESC TEC & University of Minho, Portugal.*
† *Universidade Nova de Lisboa, Portugal.*
‡ *The last two authors are with Université Catholique de Louvain, Belgium.*

using (mainly cheap) commodity hardware and whose infrastructures can be shared by a multitude of tenants [1].

The cloud computing model gained a lot of attention due to its pay-per-use business model which allows medium and small sized companies to grow and shrink their information processing infrastructure on a daily basis. This feature is usually referred to as cloud *elasticity*. It is essentially achieved through dynamic resource management: exposing the physical resources (i.e., equipment) as virtual resources and making them available to the users at fine or coarse granularity to fit their particular (dynamic) needs (e.g., a user can transparently allocate a number of CPUs or storage disks that can belong to one or several physical computers); since this mechanism is seamless to the user, the latter can demand or release virtual resources without having to take into consideration the physical resources that are hidden below.

The Limits of the Cloud Computing Model

Unfortunately, the cloud computing model is becoming increasing challenged to collectively guarantee the "everywhere, anytime" properties due to the exponential growth of data generation and processing requirements of many emergent applications, such as applications in the realm of the Internet of Things (IoT), that produce large amounts of data that have to be processed within useful, and sometimes small, time windows.

According to IBM, 90% of the data on Earth has been created in the last two years [2]; the IDC analysis [3] estimates that IoT workloads will increase nearly 750% between 2014 and 2019, generated by tens of billions of smart devices. Since the major part of this data is often stored and processed at the data centers (as described in Fig. 1), many concerns arise about how these extreme volumes of data should be transported, stored, processed, maintained, and made available.

Resorting to cloud infrastructures to support such applications is infeasible not only due to their unprecedented scale and requirements, but also due to the need to transfer large amounts of data to the cloud, which can lead to unbearable delays in the processing of data. Additionally, many of the components associated with IoT applications often experience weak or intermittent connectivity, which can lead to availability issues if the logic and storage of these applications is fully delegated to a remote cloud infrastructure.

The Time for Edge Computing

The Edge Computing model (depicted in Figure 2) was proposed to complement the current cloud computing model by moving some storage and computations to the "edge of the network" [4]. The essence of this proposal is that the majority (more than 90%) of "raw data" generated by IoT and stored at the cloud data center is arguably useless; in fact, it was observed that aggregates of data are often sufficient for most applications [5].

In the edge computing model, edge nodes with sufficient aggregate storage and computational capacities are placed closer to the end user,

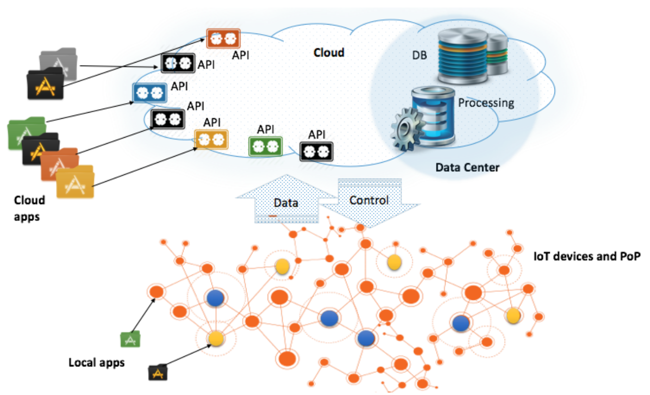

Figure 1: The cloud computing model.

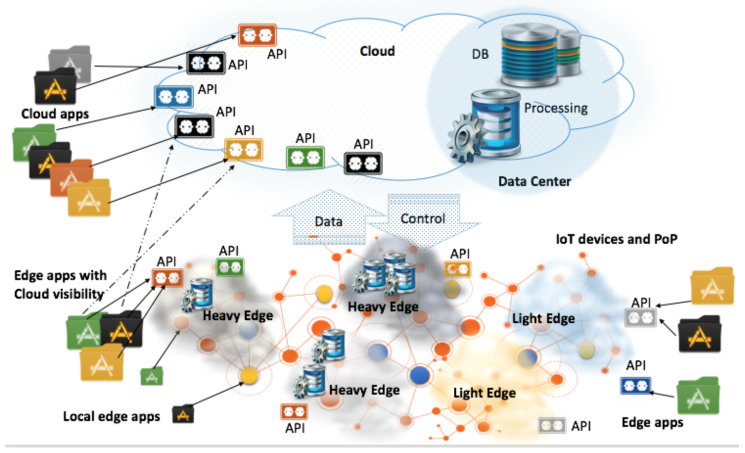

Figure 2: The edge computing model.

i.e., at the logical extreme of the network where the application is deployed. These nodes are then leveraged to offload a significant fraction of storage and computing requirements of applications, whereas complementary processing and storage can be performed at the cloud data center if necessary.

This model brings significant benefits. From a data-centric viewpoint, these benefits can be summarized as follows:

- reducing the amount of data handled (stored, processed, and maintained) at the data centers, which also leads to lower costs in using cloud infrastructures;
- reducing the network load by decreasing the data to transport by processing and aggregating it at the edge;
- improving the user experience as data processing is being performed

closer to the user, which enables lower response time especially when the data center is not reachable (this however has a potential to lower the quality of responses provided by users due to computations progressing with only partial view of the data);

- improving data security/privacy since the data that might be considered sensitive for the user is never exposed to the cloud infrastructure, a target which is easily accessible for attackers and that has increased gain;
- additionally, by storing data locally, such data is easily available to enable local decision taking, which can improve availability and overall performance/quality of applications.

Challenges at the Edge

Edge networks are usually composed of large sets of heterogeneous, loosely coupled computing nodes. Because of this property, edge applications will be frequently exposed to several aspects that make their design and execution highly challenging, namely high membership dynamics (nodes joining, leaving, and becoming unavailable), intermittent connectivity (frequent partitions), and weak communication ordering (message delivery may not respect the order in which messages were originally sent).

In fact, many distributed applications currently are designed to operate under strong consistency models, using solutions such as replicated state machines. This happens due to the complexity in designing applications under weaker consistency models. Unfortunately, the use of such techniques is incompatible with the properties enunciated above. Therefore, there is a clear need to rethink how to develop distributed applications to run on edge computing environments.

No current solutions exist that enable the easy development and deployment of applications under the edge computing paradigm. There is a lack of support for the execution of general computations in such a challenging environment. In particular, the current state of the art for performing computations on edge networks consists of gossip and peer-to-peer algorithms, which are able to do aggregate computations and provide content distribution of immutable data across the edge. Unfortunately, these techniques are still restricted in terms of scale and they are not easily extended to support general computations on the edge i.e., to perform any form of distributed computation over shared and mutable state.

Directions for Edge Computing

In this section we cover multiple relevant aspects of current technologies and solutions that can benefit the emergence of edge computing as a standard paradigm to develop efficient, robust, and useful applications. Furthermore, we identify aspects on current solutions that limit current solutions and that have to be addressed.

Hardware

Although edge computing technology is relatively new, it is attracting both the academic and industrial communities. A solution that has been suggested by leading companies such as Cisco, IBM, and Dell is to develop special hardware like servers, routers, and boxes, that can serve as edge computing devices. Dedicated edge computing devices are already in the market, e.g., Cisco UCS E Series Servers, Cisco IOx software, and Dell Edge Gateway 5000 Series, in addition to proxies, mobiles, motes, routers, etc., that can also serve as edge devices as well. Although the above edge devices enable many edge application scenarios, they are often used as stand-alone nodes that are directly connected to the node running the application and/or to the cloud data center.

However, this direction is far from exploiting the entire power of an edge network, in particular it does not enable collaborative storage and computations directly among edge devices. To achieve the full potential of the edge computing paradigm, an active effort must be conducted to *develop hardware that enable and simplify the management of direct interactions among devices executing edge-computing applications.*

Additionally, using mobiles as edge devices is currently limited to media streaming and immutable data sharing[1], and therefore lacks the ability to support general-purpose edge computing techniques that can be conducted in existing edge hardware, as presented above. This happens due to limited connectivity and capabilities on mobile devices. This implies that an effort has to be made to *develop mechanisms that can enable the participation of highly heterogeneous devices in generic edge computations*, in particular by exploiting mechanisms that are synchronization-free as to ensure availability even in scenarios where connectivity is restricted.

Cloud Computing Virtualization

Cloud computing can be seen from two different perspectives: (1) a technical perspective that deals with virtualization technology and resource management to form a large common infrastructural backbone, and (2) a business perspective that exploits a large number of aggregated resources and support infrastructure to provide services, hence the terms XaaS (Everything as a Service) [1].

From the technical perspective, virtualization cannot take advantage of the application semantics to improve performance. The reason is that virtualization aims to form a single backbone layer on top of which services can be built thus removing the burden of dealing with the inherent complexities of the underlying environment. Although this is useful to many applications, it can be limiting to others due to the latency issues that can arise from abstracting the low level aspects of a system design, deployment, and operation.

In fact, the need to explicitly deal with low-level aspects, such as high communication latency or network partitioning, is not exclusive to edge

[1]You may want to have a look at recent research projects like FP7 Mobile Cloud Networking (`http://www.mobile-cloud-networking.eu/site/`), FP7 Mobile Cloud (`http://www.fp7-mobilecloud.eu/`), and CMU Hyrax (`http://hyrax.dcc.fc.up.pt/`).

computing. It also exists in classical cloud computing. For instance, driven by the CAP theorem [6], it has been shown that many applications favor trading (strong) consistency for availability to improve user experience as those from the companies SoundCloud, Bet 365, Riot Games, Rovio, and Trifork which favored technology that avoids strong synchronization between components of a large-scale distributed system, either in terms of replication protocols employed, or through the use of specialized datatypes that can be operated in a safe way independently by each system component [7][2].

This recent trend in the design of applications and storage solutions in the context of large-scale cloud computing-based systems denotes a clear need to *invest in developing mechanisms to support the operation of edge-computing applications that require strong synchronization among its components.* Meaning that in an environment that is loosely coupled and highly asynchronous, one should favor *synchronization-free* solutions.

From a business-centric perspective, the cloud computing paradigm has found significant success from its ability to be easily exploited in the context of XaaS. A robust and efficient platform for performing edge-computation and the mechanisms required to support the operation of such an architecture have therefore to be further advanced as to *enable their exploitation and availability to a broad class of consumers as an "Edge Computing as a Service (ECaaS)".* In our view this is an essential step to achieve the full potential of this computational model.

Synchronization-Free Programming

We present briefly the main ideas of synchronization-free programming and explain how they fit edge computing [7, 8]. Traditional approaches to distributed system design do not generalize to edge networks because they depend on strong synchronization between multiple parties, such as the ones provided by uniform consensus. Unfortunately, such strong synchronization primitives are impossible to achieve in a scalable fashion on edge networks. An alternative to the use of such primitives is to rely on synchronization-free programming.

Consider a distributed data structure replicated over n nodes to improve robustness in the case of node failures. Each node has a copy of the data structures current value. When an operation is invoked at one node, a new value is computed and all nodes must be updated to the new value. This must be done consistently in the face of concurrent operations, node failures, and network disruptions ranging from variable latency, to message drop, or partitions.

Leveraging strong synchronization primitives, one would resort to a solution based on an uniform consensus algorithm such as Multi-Paxos [9] or Raft [10]. Many industrial systems use this solution, e.g, Google's Chubby Lock Service uses Multi-Paxos. In this solution, the consensus algorithm is executed across all n nodes for each operation, which besides being expensive and non-scalable, also does not provide availability in face

[2]These novel approaches have been explored in the context of a previous European research project called SyncFree (`https://syncfree.lip6.fr`).

of significant node failure or disruptions on the network.

The alternative proposed in the context of synchronization-free programming is based on the observation that, in most cases consensus is not strictly necessary for maintaining replicated data structures. Replicated data structures can be achieved using a Conflict-free Replicated Data Type (CRDT) [7]. A CRDT satisfies the mathematical property of Strong Eventual Consistency (SEC), which guarantees that replicas are consistent as soon as they observe and execute the same set of operations.

This enables the programming of distributed applications that do not have to be concerned explicitly about synchronization among components that might be loosely coupled. Instead, it becomes enough to ensure that *eventually* different components of the system are able to communicate with each other and exchange synchronization information, which is much weaker than uniform consensus.

This approach works well and is being used in several industrial systems, such as applications from the companies SoundCloud, Bet 365, Riot Games, Rovio, and Trifork. And its success points toward the need to *further exploit synchronization-free programming models to develop robust and available applications under the edge-computing model.*

Hybrid Gossip

Gossip is an effective approach for implementing aggregate computations on highly dynamic networks. In its most simple materialization, in a gossip algorithm, each component of a system will periodically interact with another randomly selected component (e.g, components in this context might represent different nodes of a distributed systems), where both can exchange information about their local state (and potentially merge it). Applications of gossip also include managing the membership of large-scale system as well as overlay (i.e, logical) network topologies [11], in a way that is both robust and scalable.

In fact, gossip-based approaches have been shown to be highly resilient to network faults, due to the inherent redundancy that its part of the design of gossip protocols. Unfortunately, this redundancy also leads to significant efficiency penalties.

Hybrid gossip solutions [11] were proposed to address this challenge by relying on the feedback produced by previous gossip interactions among nodes, such that an effective and non-redundant structure of communication can naturally emerge among nodes. This communication structure depends on the computation being performed by nodes, and it enables nodes to fundamentally communicate and coordinate through this structure, lowering the redundant communication among them.

The communication paths remaining among nodes are used to convey minimal control information, which enables the system to detect (and recover) from failures that might affect the emergent structure, as well as enabling nodes to fall back to a pure gossip strategy when a large number of failures (node crashes or network failures) happen.

The hybrid gossip approach has been introduced in [11]. Plumtree in particular, is used in industry: the Basho Riak database uses it to manage the underlying structure of its ring topology which is used to map data object keys into nodes (through consistent-hashing).

As discussed above, hybrid gossip solutions are efficient in stable scenarios and sacrifice efficiency for robustness in face of highly dynamic environments. Edge-computing environments, due to their scale and loosely coupled nature, are typically highly dynamic. This leads to the need for *further exploring hybrid gossip protocols* that can be tailored into supporting large-scale applications running in between the edge and the central infrastructure offered by data centers, in a robust and efficient manner.

Lasp and Selective Hearing

Lasp is a synchronization-free programming model that uses CRDTs as its primary data abstraction [12]. Lasp allows programmers to build applications through composition of CRDTs, while ensuring that the result of the composition also observes the same strong convergence properties (SEC) as the individual CRDTs that make up the composition. Lasp achieves this by ensuring that the monotonic state of each object maintains a homomorphism with the program state during its execution [12].

Lasp provides many benefits for developers of distributed applications; no longer do developers need to reason about the ordering of events or concurrent operations that need to be protected by locks or mutexes, as computations written in Lasp are guaranteed to have deterministic execution with minimal amounts of coordination. However, Lasp does this at a penalty to the application developer; specifically, the reduction of coordination comes at the cost of increased state. Lasp takes this view because supporting large amounts of clients that will operate under the conditions required by edge networks results in traditional solutions that rely heavily on event ordering, such as Paxos or Raft, intractable.

However, for the Lasp programming model to be successful, it must be paired with a scalable runtime system that allows for efficient, fault-tolerant event dissemination of replicated state in large-scale networks. The Selective Hearing model, which pairs the Hybrid Gossip approach and the Lasp mechanism for composition of CRDTs, provides just that: Lasp's tolerance to message reordering and duplication can exploit highly-efficient protocols for state dissemination that guarantee weak event ordering.

For supporting the operation of applications in environments with no guarantees on event ordering one has to *further exploit models such as Lasp and Selective Hearing*, which enable to define applications that do not make any assumptions on ordering of events, without putting a significant burden on the reasoning of developers.

LightKone: An Upcoming H2020 Project on Edge Computing

LightKone is a new European Horizon 2020 project that will develop novel solutions and technology for defining and executing general-purpose computations directly on an edge network without the need for a central coordinator.

The project plans to achieve this by combining the two previously discussed approaches: synchronization-free programming, where theoretically sound approaches are used to ensure data convergence under replication with concurrent writers; and hybrid gossip, techniques for supporting direct collaboration and communication efficiency through the use of emergent structures that self-adapt to dynamic environments.

Although both of these techniques are already used separately in industry, combining them addresses a specific tension: hybrid gossip protocols support efficient and resilient communication in an edge network, however provide weak ordering guarantees; whereas synchronization-free programming supports distributed programming with shared state without the non-determinism that typically arises from protocols with weak ordering.

Therefore, we expect this combination to lead to an useful substrate for large-scale edge programming that is tolerant to intermittent connectivity (frequent partitions), high membership dynamics (nodes joining, leaving, and failing), and weak ordering of communication channels.

How Does it Work?

We briefly explain how general-purpose edge computation will be achieved by combining synchronization-free programming and hybrid gossip.

Both the data and the computation are spread across all nodes. Each node contains just a small part of the global state and that state is allowed to exist on multiple nodes for resilience (replication).

Each node is responsible for a small fraction of computation over the state that is locally accessible, and multicasts the result to nodes whose computations require that result. When a node receives results from other nodes, it merges the results with its own data. The merge operation does a monotonic update of the local data. All nodes repeat continuously the three operations receive data, compute, and send results.

Due to the formal properties of synchronization-free programming, namely monotonicity and strong eventual consistency, this allows to advance the global computation in a coherent fashion without resorting to other communication or synchronization mechanisms. Additionally this approach is resilient even in face of node or communication failure. The Selective Hearing prototype that is based on this idea shows the soundness of this approach.

Light Edge and Heavy Edge

The widely different applications that we are targeting impose very different requirements on the underlying infrastructure. For example, IoT applications must carefully manage energy consumption of sensor devices and keep them in low-power mode as much as possible. A edge-centric data center application, on the other hand, must carefully manage communication between thin clients and data centers, so that clients are not hindered by latency and staleness.

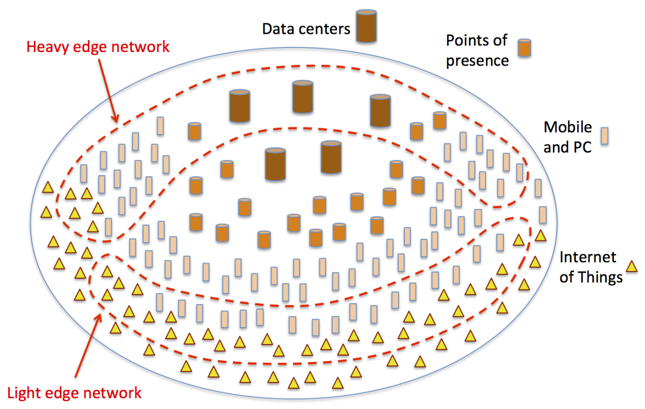

Figure 3: Weight of an edge network: light edge networks have many small nodes only, heavy edge networks also have some large nodes.

Yet there is much in common between these extremes, since all run in decentralized fashion across many nodes. For example, both IoT applications and edge-centric data center applications can allow small devices to communicate with each other, exchanging information and performing (partial) computations locally.

To distinguish the two qualitatively different kinds of edge networks, we introduce the concept of the *weight* of an edge network. We define a *light* edge network as one that contains only small nodes with limited resources (no data centers or points of presence).

We define a *heavy* edge network as one that contains many small nodes, as before, but also a small number of large nodes (data centers or points of presence). Figure 3 illustrates the two cases. The qualitative problems facing these two cases are quite different: light edge networks must do fine-grained resource management, and heavy edge networks must balance the work between the large nodes and the small nodes.

Concluding Remarks

The ever-growing amounts of data generated over the Internet are raising real challenges for the Cloud Computing model that often delegates most of its computation and storage towards the cloud data center. The Edge/Fog Computing model can be seen as a distributed extension of the Cloud model through breaking down the core of the cloud into a network of smaller clouds (i.e., the Fog) often located at the proximity of the user, thus bringing several performance benefits and novel classes of applications.

In this article, we survey the state-of-the-art of Edge Computing and recent advances in computation and communication showing how it falls short of enabling a wide range of applications in which mutable states can be used, without compromising the application semantics or performance.

Our prospective project (i.e., LightKone) will exploit recent advances in synchronization-free computations (that support shared mutable states with high availability and eventual consistency under hostile communication mediums) and an underlying hybrid gossip dissemination layer (that is based on tree-based gossip which combines lightweight and robust gossip paradigms, subject to the network reliability conditions). The project will also address the related challenges of security, scalability, availability, and sustainability, forming a comprehensive theoretical and practical solution.

One of the goals of the project is making computations at the edge reality. Consequently, the involvement of the industrial partners of the project will be key to analyze, suggest, and develop new paradigms, applications, and APIs that can be exploited at the edge. We are optimistic towards making this technology a standard that advances hand in hand with the current solutions, therefore exploiting the best of both worlds: immutable and shared mutable state computations at the edge.

References

[1] M. Armbrust et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, p. 50, apr 2010.

[2] IBM, "Bringing big data to the enterprise," 2016.

[3] International Data Corporation, "Datacenter Investments Critical to Internet of Things Expansion, IDC Says," 2015.

[4] F. Bonomi et al. , "Fog Computing and Its Role in the Internet of Things," *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, 2012.

[5] Steve Wexler, "Dell Does IoT… Totally!," 2015.

[6] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News*, vol. 33, pp. 51–59, June 2002.

[7] M. Shapiro et al., "Conflict-free replicated data types," Tech. Rep. RR-7687, INRIA, July 2011.

[8] P. S. Almeida, A. Shoker, and C. Baquero, "Efficient state-based crdts by delta-mutation," in *Networked Systems - Third International Conference, NETYS 2015, Agadir, Morocco, May 13-15, 2015, Revised Selected Papers* (A. Bouajjani and H. Fauconnier, eds.), vol. 9466 of *Lecture Notes in Computer Science*, pp. 62–76, Springer, 2015.

[9] T. Chandra, R. Griesemer, and J. Redstone, "Paxos made live – an engineering perspective," in *PODC '07*, Aug. 2007.

[10] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of ATC '14, USENIX Annual Technical Conference*, 2014.

[11] J. Leitão, J. Pereira, and L. Rodrigues, "Epidemic broadcast trees," in *Symp. on Reliable Dist. Sys. (SRDS)*, pp. 301–310, Oct. 2007.

[12] C. Meiklejohn and P. Van Roy, “Lasp: A language for distributed, coordination-free programming,” in *Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming (PPDP 2015)*, pp. 184–195, Assoc. for Computing Machinery, July 2015.