Distributed Cloud Computing: Applications, Status Quo, and Challenges

Report on Dagstuhl Seminar 15072

Yvonne Coady
University of Victoria

Oliver Hohlfeld
RWTH Aachen University

James Kempf
Ericsson San Jose

Rick McGeer
CDG, SAP America & US Ignite

Stefan Schmid
TU Berlin & T-Labs

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

A distributed cloud connecting multiple, geographically distributed and smaller datacenters, can be an attractive alternative to today's massive, centralized datacenters. A distributed cloud can reduce communication overheads, costs, and latencies by offering nearby computation and storage resources. Better data locality can also improve privacy. In this paper, we revisit the vision of distributed cloud computing, and identify different use cases as well as research challenges. This article is based on the Dagstuhl Seminar on Distributed Cloud Computing, which took place in February 2015 at Schloss Dagstuhl.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Cloud Computing*

Keywords

Distributed Cloud Computing, Distributed Systems, Dagstuhl Seminar

1. INTRODUCTION

Most of the focus in public cloud computing technology over the last 10 years has been on deploying massive, centralized datacenters with thousands or hundreds of thousands of servers. These datacenters are typically replicated with a few instances on a continent wide scale in semi-autonomous zones. Even though this model has been proven quite successful in economically scaling cloud services, it also has drawbacks. For instance, the failure of a zone can lead to a service dropout for tenants if the tenants do not replicate their services across zones. Some applications may also need finer grained control over network latency than it is currently provided by a connection to a large centralized datacenter. An application may further benefit from being able to specify location as a parameter in their deployment. Nontechnical issues, such as the availability of real estate, power, and bandwidth for a large mega datacenter, also enter into consideration. Being forced to use a remote datacenter may also be problematic in terms of privacy and legal restrictions, e.g., in the presence of incongruent political regimes.

An alternative model is to have many micro datacenters, each interconnected by medium to high bandwidth links and managed as the distributed datacenters were one larger datacenter. This distributed cloud model is perhaps a better match for private enterprise clouds, which tend to be smaller than the large, public mega datacenters. It is also an attractive solution for public clouds run by telecom carriers which have facilities in various cities with power, cooling, and bandwidth already available. It may be particularly appropriate for Europe, with a multiplicity of governments, cultures, and stakeholders. It is further attractive for mobile operators, since it provides a platform whereby applications can be deployed and easily managed. Such applications can then benefit from a tighter coupling to the wireless access network.

A third model is a grassroots distributed cloud as bottom-up collaborative effort from many small devices, including personal computers, tablets, cellphones, and small infrastructure elements such as programmable wireless base stations. This personal-devices cloud bears the same relationship to the centralized cloud that efforts like BOINC [3] have to supercomputing and infrastructure Grid computing. For a number of the use cases mentioned below, these wide-area personal-device distributed clouds have a significant role to play. Early examples that show the potential of distributed clouds in practice include the Seattle testbed [21] and Project BISmark [2].

Also note that the distributed and massively centralized models are not mutually exclusive, e.g., a public cloud operator with many large datacenters distributed internationally could manage the network of datacenters like a distributed cloud. The distributed cloud model also encompasses the federated cloud model, where datacenters are managed by different organizations that federate to allow users to utilize any of the connected datacenters. One distinguishing factor between the federated cloud and the more tightly coupled distributed cloud model is whether authentication is handled centrally or whether each datacenter handles authentication individually, with authentication for entry into the distributed cloud implemented using single sign-on.

In many scenarios distributed clouds are already a fact of life that must be dealt with. They arise by the desire for federation of datasets and computational resources, especially in governmental or corporate institutions that are

internally fractioned into different departments, divisions, business units, etc. At a more global level this leads to an often heterogeneous computing infrastructure, with a need to perform non-trivial computations and accesses across internal boundaries.

In general, designing and operating a distributed cloud is a multidisciplinary research problem that involves distributed systems and databases, programming models and abstractions, and networking. Practical deployments are further challenged by the heterogeneous nature of resources and providers, and the relatively limited networking capabilities (relative to clouds based in a single datacenter). To exchange visions on how to design and manage future distributed clouds as well as to discuss a research agenda, we organized a Dagstuhl seminar. The motivation of the seminar was to bring together leading academic and industrial researchers in the area of cloud computing, distributed systems, networking, and programming languages to discuss the opportunities and challenges of distributed cloud computing. In particular, we were interested in discussing potential use cases, trade-offs between federated and integrated distributed clouds, and unique research challenges arising when designing and operating distributed clouds.

The Dagstuhl seminar on Distributed Cloud Computing [6] was held in February at Schloss Dagstuhl - Leibniz Center for Informatics [7] in Wadern, Germany. 22 researchers attended the multidisciplinary seminar. Over the course of the 3 day seminar, 15 presentations were given on various aspects of distributed cloud computing or the disciplinary areas relevant to distributed cloud. The seminar shared two talks with the concurrent seminar on Foundations of Networking [5] and attended one of the Foundations of Networking talks. Taking the presentations as input, the workshop then broke into three groups to discuss a research agenda for distributed cloud computing. The groups were asked to come up with 3 questions in their particular area (i.e., distributed systems, programming models, and cloud) and two for the other two groups. Slides, abstracts of the talks and reports from the breakout groups are available on the seminar web site [6].

The scope of this paper is to summarize the seminar outcomes. We start by outlining discussed application areas and benefits before we present research challenges.

2. APPLICATIONS & BENEFITS

There are several motivating factors for cloud computing, where one of the most compelling is reducing costs. Distributed cloud computing additionally offers redundancy and reliability, as well as geo-replication. A distributed cloud can provide instant fail-overs by having remote replicas that can be booted up immediately in case of failures.

Applications that can benefit from locality include real-time applications where latency is important (e.g., industrial applications, virtual reality, interactive collaboration, gaming, machine-to-machine communication, smart grid control), and any application where the local regulatory environment requires user account data to be stored in the same country where the user lives. Also hybrid clouds, consisting of a private datacenter and a public datacenter, can be an attractive use case: at times of peak demand, additional resources can flexibly be obtained from the public cloud, allowing enterprises to provision their datacenters for average rather than peak load, thereby saving the capital expense of building out the data center to handle the peak load. Content delivery networks are another example where storing data close to the user is required. The opposite case—moving the processing closer to the data—may become important when the data sets are large and networking costs prohibit actually moving the data.

2.1 Reducing Wide-Area Traffic

A frequent argument for distributed clouds is resource efficiency: by exploiting locally available storage and compute resources (or by moving compute resources closer to data generators), wide-area communication can be reduced or even avoided.

Examples for offloading the core network by turning the traffic around in edge distributed clouds, include sensor networks, or "things" (as in Internet-of-Things), that generate large amounts of data to be sent via wide area networks for processing. Transmitting these data streams to a remote datacenter is not only inefficient but often also pointless: often, administrators and data scientists are only interested in a very small fraction of the data. Accordingly, it is desirable to perform *in-situ* processing, exploiting nearby resources, to aggregate and filter data, focusing only on the interesting parts.

Applications for local processing are many. James Kempf gave the example of an airplane: an airplane's jet engine produces up to 500GB sensor data per flight. This data should be analyzed locally and quickly at the airport, before the airplane takes off for the next flight (deadline constrained data processing). Since the wide-area capacity might be limited to transfer such amounts of data to remote datacenters for processing, local datacenters are needed.

Similarly, Rick McGeer gave the example in the context of the networks at the LHC at Cern in which each sensor can generate millions of events in a short amount of time. Since not all events are relevant and need to be transferred to the remote scientists, a distributed network of filtering compute nodes can reduce the overall traffic volume. The yield from the ATLAS detector at LHC is about 200 events/second out of a total of 600 million generated events/second [17], a yield rate of about one in three million: the rest are filtered by a combination of hardware and software. This is an example of a centralized high-bandwidth sensor, where filtering can be done locally. However, there are also many examples of high-bandwidth distributed sensors, e.g., the Cooperative Adaptive Sensing of the Atmosphere project [4], which envisions an array of relatively closely-spaced weather radars across the tornado belt, each of which can generate over 150 Mb/s. These examples motivate scenarios for using specialized and dedicated infrastructures for big data processing that are limited to a particular set of users.

A more futuristic example requiring a more generic distributed compute infrastructure was been given by Tim Wood: augmented reality applications may require much local compute power, e.g., in order to process and render information on the user's glasses: if the user's devices do not have enough capacity to perform this task, it may be out-sourced to a nearby cloud.

While it is beneficial to move data processing nodes closer to traffic generators or storage systems, not every node can be used to perform compute jobs. In this respect, Lars Eggert emphasized that moving VMs to storage systems is challenging since their CPU capacity is limited and will then be

occupied by compute jobs. In these cases, it can be beneficial to still move the data to dedicated compute clusters for processing when the data is hot. The processing pipeline should focus on hot data rather than cold data. Lars discussed further aspects of storage systems and their evolution in his talk on data management.

A general perspective on challenges arising in big data processing in distributed clouds was presented by Patrick Eugster. His talk was motivated by two observations: first, copying all data to a single datacenter for subsequent analysis is inefficient, if feasible at all under sharing regulations. Second, especially when leveraging public clouds, processing confidential data in the cloud is undesirable with the security dangers implied by multi-tenancy underlying cloud platforms. His talk then presented a survey on some practical first steps towards addressing these challenges. This includes work on (a) geo-distributed big data analysis and (b) assured cloud-based big data analysis. In short, the former consists in moving computation towards data rather than only the other way around, and the latter consists in leveraging a combination of replication and partially homomorphic encryption to ensure integrity/correctness and privacy of big data analyzed in the cloud.

2.2 Reducing Latency

Another motivation for the distributed cloud is latency: today, the latency within a datacenter rack is an order of magnitude lower than across the entire datacenter, which in turn is an order of magnitude lower than the latency to the next datacenter. Using a nearby datacenter may be attractive for a group of mobile users wishing to collaborate on an interactive document, or for computer gaming. Users of a local cloud may not only be humans, but also computers, robots, or self-driving cars. One futuristic use case given by Tim Wood concerns robots seeking to coordinate their movements in an accurate and synchronous manner.

In general, quantifying the impact of reduced latency on user perception is however non-trivial. As Oliver Hohlfeld pointed out, whether service quality is perceived to be good or bad depends on a multitude of user specific factors including the users' expectation and the users' context, e.g., on the current location of the user. As an example, Oliver presented his assessment of latency on gaming quality in user studies. He argued that one should take a user perspective when designing distributed clouds, where human users and machine users will differ in their requirements.

2.3 Computation at the Edge

One use case for micro datacenters is distributed Utility Computation (UC), as outlined by Jonathan Pastor. His talk on the Discovery Initiative [8, 16] proposed to leverage compute resources hosted at Points-of-Presence for bringing UC closer to end users. In light of this problem, Jonathan presented LUC-OS: a fully distributed IaaS system, based on P2P architecture. The architecture targets scalability by managing hundred thousands of VMs upon thousands of physical machines spread throughout hundreds of sites.

Hagen Woesner presented a second motivation for moving distributed cloud computing to the edge. He raised the question of which applications we see centralized and which ones distributed, given infinite bandwidth to the home. He argued that customer premise equipment is the most suitable location for functions concerning traffic monitoring, QoS, or security. He also argued that it could be beneficial to push the energy cost of computation to the end user, where, e.g., renewable energy can be exploited. His talk also opened the question on how resources should be exposed to an orchestration platform. One potential solution would be to expose them as networking function forwarding graphs.

Chip Elliott also pointed out that a telco-driven distributed cloud à la EC2 can be an attractive business model: users may be willing to pay more for a service which ensures data locality.

2.4 Education

Many discussions revolved around the potential of using distributed cloud computing in education. This use case is appealing since it can provide easy access and early experiences on programming distributed applications, as well as on distributed state management (e.g., by applying consistency models).

On this regard, Justin Cappos introduced the Seattle distributed cloud testbed [21] (also available via the SIGCOMM Educational Resources [23]) and showcased its potential for educational usage. Justin presented several Seattle based programming assignments for in class usage (see [22]). He further commented on his experience on the educational appeal of Seattle. Seattle simplifies experimentation with a distributed networked systems. Reporting on past experience, it is very appealing even to high school students to test their networked programs by running them at remote computers, e.g., phones located in China.

This perspective was complemented by Rick McGeer's discussion on educational games that run in a distributed cloud. This use case is motivated by using the cloud to avoid sharing educational game code with the end users to 1) prevent cheating / maintain control and to 2) simplify portability: by only streaming a video rendered by the game over the network, fewer platforms need to be supported by game vendors.

3. RESEARCH CHALLENGES

Distributed cloud computing comes with many challenges. Distributed clouds will seldom be designed from scratch, but grow out of existing infrastructures. Moreover, how to best allocate resources as well as to program a distributed cloud, also depends on the context, the specific application, and the objectives.

Initial motivational questions. Around the seminar talks, we organized panels, breakout sessions and standup meetings to identify and discuss enablers as well as challenges of the vision of distributed clouds. For example, we discussed questions such as:

1. Deployment models for cloud: hyper-centralized (e.g., one datacenter per continent), centralized (two or three datacenters per continent), distributed (one datacenter per metro area), hyper-distributed (a datacenter on your street corner). In what cases is a specific deployment model favored?

2. Federated vs. integrated vs. autonomous: what are the trade-offs of these distributed cloud models? Distributed operating system vs distributed cloud management (e.g. distributed OpenStack) approaches: what are the advantages and disadvantages of each?

Service composition and decomposition: what are the primitives for infrastructure management?

3. How to support for network virtualization going out of the datacenter and between datacenters? What service differentiation protocols could or should be implemented in a distributed cloud? How does it compare to Google B4 [12]?

4. What are suitable programming models for wide area clouds? Which details on resources (e.g., bandwidth and latency) and location should be exposed to the programmer? What logical abstractions are useful and tolerable?

5. What consistency models are required in a distributed cloud? Can we assume synchronized environments à la Google Spanner [11]? Which distributed agreement protocols (e.g., *RAFT* [18], *Paxos* [15], and *ZooKeeper* [10]) are used in which context?

6. What benefits and challenges exist in terms of privacy and security?

7. Economics of distributed cloud deployment: cheaper or more expensive than centralized cloud? What are the costs of networking resources for distributed cloud vs. centralized cloud?

8. How can academia contribute what companies cannot? How to conduct scientific experiments on the distributed cloud: scalability and repeatability of experiments?

The discussions around these questions resulted in the identification of challenges in three major lines of research, i.e., programmability, consistency, and security.

3.1 Programmability

The first research challenge concerns identifying proper programming models, primitives, and abstractions for distributed cloud computing. Today, cloud computing relies on virtual machines as execution model. However, a programming model is still missing.

Much progress has been made over the last years in the design of network programming languages providing higher levels of abstractions, see for example the *frenetic* language family by Nate Foster et al. [9]. Indeed, the software-defined networking paradigm can serve as an inspiration for distributed cloud computing as well. In his presentation, Chip Elliott used the notion of *software-defined infrastructures*, to describe the recent convergence of multi-tenant clouds, distributed clouds, network functions virtualization, and software defined networking.

There seems to exist a wide consensus that in terms of programming languages for distributed clouds, there is no *lingua franca*, no "one size fits all": different components and their interconnections will likely be programmed in different languages. In particular, it is often convenient to use constrained, domain-specific languages to provide the right, hopefully high-level abstractions, to allow programmers to focus on the important concepts. Often times these also introduce verifiability: in contrast to Turing complete languages, programs can be checked efficiently. However, there is also a wide consensus among the participants, that there is still a long way ahead, in terms of developing more powerful programming languages for networked systems.

Moreover, while programming a network or infrastructure from a "logically centralized" software controller can be convenient and attractive, for its simplicity and since many operational tasks are inherently non-local, implementing correct controller software today is non-trivial. In her talk, Jennifer Rexford discussed challenges and pitfalls in programming simple notions of MAC learning switches and stateful firewalls.

An interesting question also pertains to the actual implementation of the logically centralized controller abstraction: Stefan Schmid argued that in order to ensure availability and fault-tolerance of a software-defined infrastructure, the controller should actually be distributed and redundant [14], and ideally handle latency-critical events close to where they occur [20]. The need for a distributed control can also come from administrative constraints. Stefan presented the hierarchical control architecture developed in the FP7 UNIFY project which aims to unify the programmability of cloud and carrier infrastructure [19].

Johan Eker took a look further into the future, where billions of devices will be connected to the clouds, which offer not only IT services but also mission critical services such as automation and health-care, requiring predictable latency and high availability. He argued that accordingly, a programming platform must expose cloud services and network functionalities in a simple and straightforward matter, and presented the *Calvin project* that aims at developing a programming framework for applications that spans over a heterogeneous hardware platform consisting of mobile sensors and cloud.

Igor Konnov emphasized the importance of verifying safety and liveness of the fault-tolerance mechanisms in a distributed cloud, and presented an abstraction-based model checker for threshold-based fault-tolerant algorithms [13]. The model checker implements novel parameterized verification techniques that allow one to check systems of arbitrary size, which is particularly important for the system sizes encountered in data centers and clouds.

In the breakout sessions, we also discussed the different programming language primitives needed for placement, configuration of non-functional and functional requirements, service discovery and SLA, in the context of application classes such as big data analytics, services (such as control loop, drive my car, do face matching), games, latency-sensitive applications, education, and web servers.

3.2 Consistency

Related to the programmability model and abstraction is the question of what type of distributed system we want to build or expose. In his talk, Marc Shapiro gave an overview of the design space of distributed systems. He further highlighted interesting trade-offs between the strength of a consistency model and performance. For example, it is well known that to achieve stronger consistency, multiple round-trip times are needed. Marc also argued that oftentimes, in distributed systems, researchers and practitioners end up needing less consistency than initially thought. He also confirmed that in the past, generic programming languages failed in the distributed world, and that there is no one size fits all in distributed systems: different requirements need to be supported, also in terms of programming libraries. What

is needed is an understanding of what a programmer intends to do with resources provided by distributed clouds.

Marc also pointed out that the differences between competing consistency models (such serializability, snapshot isolation, eventual consistency, etc.) can be subtle and hard to understand, but reflect fundamental trade-offs between fault tolerance, performance, and programmability (see e.g., [1]). For instance, describing consistency models in terms of acceptable histories is not very informative. Marc argued that what programmers really care about is a consistency model's *properties*: guarantees (i.e., what kind of application invariants are ensured automatically by a model), scalability (i.e., opportunities for parallelism and implementation freedoms in a model), or abstract classes of guarantees (e.g., partial-order-type invariants, equivalence-type invariants, identical-observer guarantee).

Also the placement and orchestration problem can be seen from the distributed system perspective. This perspective involves trade-offs between performance (placing components close to each other) and reliability (placing components far from each other). Distributed cloud computing enables the moving of execution of distributed applications towards client machines. Annette Bieniusa argued that current data management solutions for cloud infrastructures replicate data among several geographically distributed datacenters but lack support for managing data maintained by clients. The SwiftCloud [24] storage infrastructure for cloud environments aims to cover this gap: SwiftCloud pushes the scalability and concurrency envelope, ensuring transactional causal consistency using Conflict-Free Replicated Data Types (CRDTs). CRDTs provide higher-level object semantics, such as sets, maps, graphs and sequences, support unsynchronized concurrent updates, while provably ensuring consistency, and eschewing rollbacks. Client-side replicas are kept up to date by notifications, allowing client transactions to execute locally, both for queries and for updates.

In multi-domain distributed clouds, different parts of the cloud will be owned and operated by different organizations. This also raises the question: How will resources be described and obtained? Will there be brokers, and how does multi-domain management work? Moreover, if parts of the cloud will be battery powered, what impact will this have on the architecture in terms of reliability, consistency, etc.?

3.3 Privacy and Security

A distributed cloud may improve privacy, by keeping data local: in the neighborhood, or at least within a region or country. However, today, we are still far from enforcing this locality. We do not have mechanisms for path control, and data may travel far even to arrive at a close cloud. Also, multitenant clouds may actually *increase* the number of entities a user has to trust. Privacy often comes at a performance cost that needs to be considered in the design phase, as argued by Oliver Hohlfeld. Fostered by the wish for data locality, privacy concerns drive migrations to private clouds, e.g., running Openstack. When data needs to be processed but not stored in public clouds, storage devices owned by the data owners can be moved next to big datacenters (e.g., operated by Amazon) for fast-path access to compute clusters. This approach addresses the issue of data ownership that remains at the storage location. In general, users need to express restrictions on data locality and data processing (e.g., where and by whom data should be stored and processed). To address these needs, Oliver referenced their efforts on designing privacy aware cloud architectures within the SSICLOPS EU project.

Hannes Hartenstein pointed out that a security objective does typically not exist "in isolation", but in combination with other objectives: confidentiality *and* performance, confidentiality *and* availability etc. In his talk, he considered two use cases that both show benefits and challenges of distributed clouds, namely confidential data outsourcing and secret sharing schemes. He argued that confidentiality may be achieved based on non-colluding cloud providers and how the resulting trade-offs with performance and availability can be tuned for the cases of outsourcing databases and outsourcing strong cryptographic keys based on secret sharing schemes. He also promoted the conduction of a threat analysis on some consistency mechanisms to determine possible security issues.

Christine Morin described her vision of *ephemeral clouds*, personalized, spontaneous and transient clouds which are allocated for the duration of a collaboration, and highlighted critical security requirements. A more tightly coupled distributed cloud model may hide the locality distinctions between physical datacenters and present the distributed cloud as one single datacenter without exposing the network interconnections. In the latter model, orchestration software manages the user's view of the compute/storage/networking resources to hide locality. Such a model may be important in cases where locality is not a critical characteristic for application deployment. In other cases, however, locality should be exposed through the orchestration layer.

4. CONCLUSION

The vision behind distributed cloud computing is to utilize software as a means to aggregate compute/storage/networking resources across distributed physical datacenters. This model addresses data locality that is incorporated as design criterion for those applications that require it. It further includes achieving scalability and reliability by performing scale-outs. The scale-out model of service deployment—deploying many small instances of a service to meet demand rather than a few large instances—has proven successful for IaaS and SaaS. Distributed cloud computing applies the same scale-out model to datacenters.

This paper has reported on our Dagstuhl seminar on distributed cloud computing, which complemented our workshop series on distributed cloud computing, held together with IEEE/ACM UCC 2013, ACM SIGCOMM 2014, and ACM SIGMETRICS 2015. We believe that the seminar has been very productive and lively, and contributed to the community building. We also hope that the seminar opened new collaborations, and we will soon be able to give a better answer to Rick's most pressing question: "How do we build the thing?"

5. ATTENDEES

This workshop was held at Schloss Dagstuhl - Leibniz Center for Informatics [7] in Wadern, Germany. We would like to thank the Dagstuhl organizers, in particular Roswitha Bardohl, and all the participants for their active contributions:

- Mark Berman (BBN Technologies Cambridge, US)

- Annette Bieniusa (TU Kaiserslautern, DE)
- Justin Cappos (New York University, US)
- Yvonne Coady (University of Victoria, CA)**: Co-Chair**
- Lars Eggert (NetApp Deutschland GmbH Kirchheim, DE)
- Johan Eker (Lund University, SE)
- Chip Elliott (BBN Technologies Cambridge, US)
- Erik Elmroth (University of Umeå, SE)
- Patrick Eugster (Purdue University and TU Darmstadt, US + DE)
- Hannes Hartenstein (KIT Karlsruher Institut für Technologie, DE)
- Oliver Hohlfeld (RWTH Aachen, DE)**: Collector**
- James Kempf (Ericsson San Jose, US)**: Co-Chair**
- Igor Konnov (TU Wien, AT)
- Rick McGeer (Comunication Design Group, SAP America, and US Ignite)**: Co-Chair**
- Christine Morin (Inria, FR)
- Jörg Ott (Aalto University, FI)
- Jonathan Pastor (Inria, FR)
- Vivien Quema (Grenoble INP, FR)
- Stefan Schmid (TU Berlin & T-Labs, DE)**: Co-Chair**
- Marc Shapiro (Inria Paris-Rocquencourt and LIP6-UPMC-Sorbonne Universités, FR)
- Hagen Woesner (BISDN GmbH Berlin, DE)
- Tim Wood (George Washington University Washington, US)

6. REFERENCES

[1] M. S. Ardekani, P. Sutra, and M. Shapiro. G-DUR: A middleware for assembling, analyzing, and improving transactional protocols. In *Int. Conf. on Middleware*, 2014.

[2] Project bismark. `projectbismark.net/`.

[3] Boinc: Open-source software for volunteer computing and grid computing. `boinc.berkeley.edu`.

[4] Casa: Collaborative adaptive sensing of the atmosphere. `www.casa.umass.edu/`.

[5] Dagstuhl seminar 15071: Formal foundations for networking. `www.dagstuhl.de/15071`, 2015.

[6] Dagstuhl seminar 15072: Distributed cloud computing. `www.dagstuhl.de/15072`, 2015.

[7] Schloss dagstuhl. `http://www.dagstuhl.de`.

[8] Discovery initiative. `http://beyondtheclouds.github.io`.

[9] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: A network programming language. *SIGPLAN Not.*, 46(9):279–291, Sept. 2011.

[10] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. ZooKeeper: Wait-free coordination for internet-scale systems. In *USENIX ATC*, 2010.

[11] J. Corbett et al. Spanner: Google's globally-distributed database. In *USENIX OSDI*, 2012.

[12] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a globally-deployed software defined wan. In *SIGCOMM*, 2013.

[13] A. John, I. Konnov, U. Schmid, H. Veith, and J. Widder. Parameterized model checking of fault-tolerant distributed algorithms by abstraction. In *FMCAD*, 2013.

[14] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A distributed control platform for large-scale production networks. In *USENIX OSDI*, 2010.

[15] L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[16] A. Lèbre, P. Anedda, M. Gaggero, and F. Quesnel. DISCOVERY, Beyond the Clouds - DIStributed and COoperative framework to manage Virtual EnviRonments autonomicallY: a prospective study. In *Virtualization for High Performance Cloud Computing Workshop*, 2011.

[17] Taking a closer look at LHC. `www.lhc-closer.es/1/3/12/0`.

[18] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *USENIX ATC*, 2014.

[19] P. Skoldstrom et al. Towards unified programmability of cloud and carrier infrastructure. In *European Workshop on Software Defined Networking*, 2014.

[20] S. Schmid and J. Suomela. Exploiting locality in distributed SDN control. In *HotSDN*, 2013.

[21] Seattle: Open peer-to-peer computing. `seattle.poly.edu/`.

[22] Seattle in the classroom. `https://seattle.poly.edu/html/education.html`.

[23] ACM SIGCOMM educational resources: Seattle testbed. `http://edusigcomm.info.ucl.ac.be/Public/20100614001`.

[24] M. Zawirski, A. Bieniusa, V. Balegas, S. Duarte, C. Baquero, M. Shapiro, and N. Preguiça. SwiftCloud: Fault-tolerant geo-replication integrated all the way to the client machine. Rapp. Rech. RR-8347, Institut National de la Recherche en Informatique et Automatique (Inria), Aug. 2013.