Scalable Computing: Practice and Experience
Volume 12, Number 4, pp. 385–390. http://www.scpe.org
ISSN 1895-1767

CHALLENGES IN CLOUD COMPUTING*

KLAUS-DIETER SCHEWE† KÁROLY BÓSA‡ HARALD LAMPESBERGER§ JI MA¶ MARIAM RADY‖ AND BORIS VLEJU**

Abstract. Though cloud computing is considered mature for practical application, there is a need for more research. The identified challenges primarily concern client-cloud interaction and cloud interoperability. As to the former one, we highlight the needs of clients, contracting and legal aspects, and missing foundations as necessary fields of investigation. For the latter one clouds are considered to constitute repositories of services, so the challenge is to realize web-scale, service-oriented, distributed computing.

Key words: cloud computing, research, service clouds

AMS subject classifications. 68N99, 68Q85, 68U99

1. The Case for Research in Cloud Computing. Currently, "cloud computing" is the most often used buzzword in computing, and many providers (Amazon, Google, Microsoft, IBM, etc.) of cloud services (IaaS, SaaS, PaaS, DaaS, ...) emphasize the many benefits of outsourcing application into a (private or public) cloud. In other words, it is suggested that cloud computing represents a mature technology that is ready to be massively used. It is our conviction that that cloud computing still requires lots of fundamental research.

In particular, most of the offerings in cloud computing are provider-centric. For instance, a client (or tenant) may rent a certain piece of infrastructure, load and execute a piece of software on it, pay for the use, and leave the cloud without leaving permanent traces. Certainly, there are many computing-intensive applications, e.g. web crawling, image processing, machine learning, etc. that fit well into such a scenario. However, if we think of a multi-user database application, its usefulness decreases significantly.

First, we have to cope with interaction, which implies massive transfer of data from and to a cloud leading to a performance problem. Second, in such applications it is usually required that the use of the database is transparent to the database owner, i.e. credentials of user must be maintained by the client and not the cloud provider. Third, if several such applications are combined, the problem of cloud interoperability arises, for which the state of the art in cloud computing is not yet well prepared.

On these grounds we identified two major areas of investigation. First, our research aims at a significant improvement of client-cloud interaction. Second, we envision that cloud computing will only unfold its full potential, if the scope is widened to web-scale distributed computing.

2. Facilitation and Improvement of Client-Cloud Interaction.

2.1. The Forgotten Tenant. Among many other problems in cloud computing we identified the lack of client-orientation as a serious problem that needs to be addressed in research. This subsumes the problems of identity of tenants, access rights, adaptivity to the needs of clients, and more. For instance, in the database application scenario above it would be indispensible to keep knowledge of users and their rights in the authority of the client instead of in the cloud. The immediate consequence of such an approach is that cloud applications become hybrid and distributed, as parts of data and software will reside on promise, while others reside in the cloud.

*This paper is a short position paper describing a research view on cloud computing in accordance with the research agenda of the recently established Austrian Christian-Doppler Laboratory for Client-Centric Cloud Computing.

†Software Competence Centre Hagenberg, Hagenberg, Austria, kd.schewe@scch.at & Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, kd.schewe@cdcc.faw.jku.at

‡Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, k.bosa@cdcc.faw.jku.at

§Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, h.lampesberger@cdcc.faw.jku.at

¶Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, j.ma@cdcc.faw.jku.at

‖Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, m.rady@cdcc.faw.jku.at

**Johannes-Kepler-University Linz, Christian-Doppler Laboratory for Client-Centric Cloud Computing, Linz, Austria, b.vleju@cdcc.faw.jku.at

As adaptivity describes a property of a system to adapt itself to different contexts, we aim at adaptivity to preferences of clients, restrictions arising from channels, and specific requirements for end-devices, in particular mobile devices. Preferences of clients refer to the way they interact with cloud servers. This already applies to the selection of services, for which we need customer-specific preference rules giving weights to providers and quality of service attributes. Our aim is to investigate the full range of possible selection preferences and to come up with languages for the expression of selection preferences for clients.

Analogously, client preferences may impact on the way parts of a cloud are handled that are owned by them or for which a client has exclusive access rights. In this case preferences can affect the location and method of storage as well as access protection; clients may even have preferences for particular protocols that are to be applied in these cases. Here, we also want to investigate, which options should be made available for clients to express preferences.

Furthermore, client preferences affect the content, functionality and the presentation of selected services. We aim at extending the rewriting-based approaches to automatically adapt choices in plots [8], which at the moment are restricted to high-level specifications. In particular, this is expected to lead to adaptation algorithms that take a specification of a (composed) service and a set of preference rules as input and produce a modified service that respects the client preferences.

2.2. Contracting and Legal Aspects. The creation of applications, in which larger portions of software are services that are offered somewhere by some service provider, is not only a scientific and technological challenges, it also implies the need to handle the relationship between service providers and service users. In a service-centric system the use of a services replaces the purchase or lease of data or a software product. The use of other's hardware as a platform replaces the purchase and maintenance of hardware by clients. The use of software services also brings with it the benefit of not being bothered by new releases, bug fixes, etc., which become part of the service.

However, what can be expected by a service user requires contracts that state explicitly, what a service offers, when it will be available, which performance can be expected, which maintenance is guaranteed, which security mechanisms are in place, how privacy is guaranteed, and how much the service costs. It has to contain regulations that apply in case one of these assertions of service is violated. Such service contracts may replace licensing agreements. As such there must be different classes of contracts depending on whether individuals use the service or groups. It also has to prescribe whether the client or the provider takes care of group-specific regulations. On the other hand, contracts must also contain rules that prescribe health conditions on the side of the client, rules, the level of security and privacy the client has to guarantee when using the service.

Our aim is to investigate the full range of regulations that have to be handled in service contracts. Leaving aside purely legal regulations such as the responsible court in case of disagreement or claims, the question is, to what degree the ingredients of service contracts can become part of the service description, in which case they could become part of a QoS ontology.

2.3. Security and Privacy – What Clients Need. The first problem area in security and privacy is connected to the management of access rights. This leads to two research problems we intend to investigate. The first one is concerned with the checking of access rights, for which we intend to develop adequate methods. Though this appears to be rather straightforward, the remaining problem is to keep track of dependencies between group rights and rights of group members.

The second problem is dedicated to inferences on access rights. For this we have to take into account that ownership should imply access rights, that membership in a group should imply that access rights of the group also apply to its members, that the right to execute a service operation should imply the right to execute the underlying view, that the right to use a composed service should imply the right to use the component services, etc.

More generally, access rights can be considered as permissions in a deontic logic. Therefore, we believe it is advisable to investigate not only permissions, but also prohibitions, obligations, and triggered actions. Thus, we aim at embedding access rights into a complex deontic logic and to study inferences in that logic. The goal s to ensure that all implied access rights must be explicitly granted. A particular difficulty arises from rights to grant access and revoke rights.

With respect to privacy tenants have to be protected against malicious users as well as against the cloud provider. With respect to the first of these hazards we pick up on the idea of specification of secrets, i.e. for data stored on a cloud it has to be specified who is permitted to see the data and who is not. Therefore, we

first have to investigate data models in more detail in order to be able to identify at which granularity level such secret specifications should be applied. For instance, in case data is organised in relations, secrets may apply to records, clusters of records, or even individual attributes. For tree-based data organisation, e.g. in case XML is used as the data model, secrets might apply to subtrees rooted at particular elements, leaf elements and attributes, or aggregated tree portions that are defined by some query expression or algebra term.

Next we have to be aware that secrets may nonetheless be discovered by means of inferences. Therefore, we investigate, which queries or sequences of queries would be necessary to retrieve a secret that cannot be queried directly. We intend to also distinguish between exact detection of a secret, detection with a tolerable error, and detection with a high probability. These results should give an indication which access rights may need to be restricted to exclude the detection (or almost detection) of secrets by means of inferences. We will also investigate the alternatives of blocking certain queries, if the result could be used to discover secrets and the application of "lying strategies". However, also the rejection of queries and inconsistency of query results can be used as valuable information inferences can be based upon.

The second problem is somehow inverse, as we want to allow a customer to retrieve some data from a cloud – access rights are assumed to be granted – without being able to see the actual query. It is well known that anonymity can only be guaranteed in an efficient way, if data is replicated. We therefore intend to study replication strategies and develop algorithms for query execution that preserve anonymity.

2.4. Epistemiological and Formal Foundations. Besides the forgotten tenant there is a serious lack of formal foundations in cloud computing starting from the simple fact that key notions such as service are not defined. Therefore, we address the fundamental research question how a uniform formal model for clouds must look like without any bias to particular languages and technology. We further investigate what a cloud has to offer to enable effective and efficient search for services as well as effective and efficient of multiple access to services by multiple tenants. The basic research component will build upon our previous research on Abstract State Services (AS^2s) described above, which has to be extended in various ways.

According to our understanding a cloud is primarily a repository of services, so we first have to specify the notion of service. The model of Abstact State Services (AS^2s) [5] starts from the assumption that a service should combine a hidden part, and a public part that is exported to service users. The hidden part is assumed to be a database-based transactional system, for which a universal model of database transformations [9] is adopted. The visual public part is represented by a collection of views, each of which is extended by a set of transaction-oriented service operations that link to the hidden database part. The whole model adopts the theory of Abstract State Machines (ASMs) [3], in particular referring to the so-called ASM thesis [4, 2].

3. Cloud Interoperability.

3.1. Clouds as Repositories of Services. Roughly speaking the research topics presented in the previous section address participation barriers in cloud computing. Clients, i.e. companies will only engage in cloud computing, if they have full control of their "outsourced" data and software, they can leave the cloud and engage elsewhere at any time, and the clent-cloud interaction is secure and reliable. There have to be formal and contractual guarantees for all this.

Nonetheless, even with all problems disussed so far being solved there remain risks, inless there is a trusted third party involved. If such a trusted party does not exist, we believe that distribution andreplication will still offer possibilities to remove participation barriers. Anyway, securing availability of cloud services will require some form of replication.

Furthermore, we argue that cloud computing can only unfold its complete strength, if services from different cloud are exploited in an interoperable way. With this in mind let us first look back at the services typically offered by a cloud.

- *Infrastructure as a Service* (IaaS) is the simplest form of cloud computing emphasizing mainly the use of hardware by the tenants. The costs are based on actual usage rather than on a priori fixed payment model. A known example is Amazon's elastic cloud (EC2).
- *Platform as a Service* (PaaS) makes computing platforms available to tenants, thus parts or all of the software of tenants are outsourced to a cloud, which is maintained by some cloud provider. The provider takes care of maintenance of hard- and software, and thus guarantees a smooth running of the tenants' applications, and the tenants pay for this service. In particular, ad-hoc usage of service components is part of the model. Examples are Microsoft Azure and Google's AppEngine.

- *Software as a Service* (SaaS) emphasizes that the services provided are in fact software indicating a shift from software licensing to leasing. The software can be made available through web services [1].
- Similarly, *Data as a Service* (DaaS) emphasizes that the services provided are in the form of data. This facet, however, is not well present in actual cloud computing offers.

We propose to look at this "cloud stack" from the angle of ownership and usage rights (and obligations). In the IaaS model the basic (hardware) infrastructure is owned by the cloud provider and leased to the client on the grounds of some cloud usage contract. Every piece of software uploaded to the leased cloud infrastructure, however, is owned by the client. So, at the end the client will build up software and data services on the cloud. In most cases such services will be used only by the client who owns these services, but it is no problem to assume that services are also made available to be used by others, in the extreme case by everyone. In this sense the client of the cloud would also become a service provider.

Similarly, in the PaaS model the basic infrastructure plus development platforms (i.e. more than in the IaaS model) are owned by the cloud provider, but again this is used by the client to build up services to be used by himself or any other (defined) larger community. Finally, in the PaaS (and similarly the DaaS) model everything on the cloud is owned by the provider, but used by the client.

Thus, looking at the "cloud stack" from this angle turns a cloud into a repository of (data and software) services, each of which have an owner, a community of users and many regulations regarding rights and obligations of providers and users. The orthogonal classificaton into private and public clouds does not change the view, it only impacts on the way the cloud is organised as a repository of services.

We therefore like to address the interoperability of clouds by means of looking at such repositories of services. In [5] a formal model – called Abstract State Service (AS^2) – for (data and software) services was developed, which can serve as a basis for these investigations. Without repeating formal details here an AS^2 is composed of two layers: a hidden data layer and a view layer on top of it. Both layers combine static and dynamic aspects. Data services are formalised by views, which in the extreme case could be empty to capture pure functional services.

3.2. Service Clouds. In [6] this formal model was extended to a formal model of a "service cloud", which combines several AS^2s with an ontology that describes them "semantically", i.e. some form of description logic is exploited to describe each service by a combination of three different aspects. Without going into formal details a service description comprises:

- a *functional* description of input- and output types as well as pre- and post-conditions telling in technical terms, what the service operation will do,
- a *categorical* description by inter-related keywords telling what the service operation does by using common terminology of the application area, and
- a *quality of service* (QoS) description of non-functional properties such as availability, response time, cost, rights, obligations, etc.

The functional description is needed to actually use the service, once it has been selected, while the categorical description is needed to locate candidate services. The QoS description is not needed for service discovery, but can be exploited to select among alternatives.

The reason for the use of description logics (since its very beginnings over 30 years ago) is that they enable the definition of concepts by necessary and sufficient conditions, and the logics are kept so simple that classification, i.e. determining subsumption relationships, is decidable. Thus, a search requires a definition of the service sought by means of a complex concept. The well-known classification algorithms for description logics then can be used to determine all instances (in the ABox) matching the complex concept.

The idea of a service ontology is already omnipresent is the web services community. Languages such as WSDL [12], OWL [11] and UDDI [7] have been introduced to capture description, publishing and search of services. However, we like to remark that whatever the description in an ontology looks like, it is indispensible that a high-level description of the service (e.g. by means of ASMs as exploited in [5]) is made available as well so that a service interpreter can be used to check the suitability of a service.

3.3. Web-Scale Distributed Computing. Service clouds can be used to build up new applications that are composed mainly out of available services. In this way we obtain distributed systems. The interaction between different components could be supported by protocols such as SOAP [10]. We argue that building such web-scale distributed applications (as illustrated in Figure 3.1 and making them again available as services present a real challenge for cloud computing and web services.

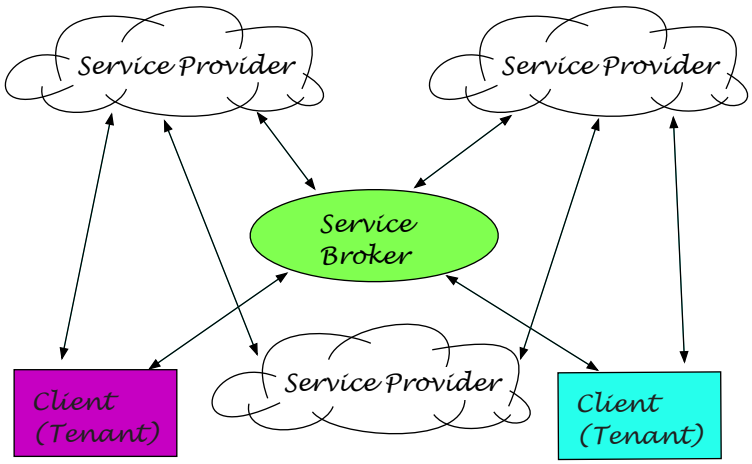

Fig. 3.1: Clouds in Web-Scale Distributed Systems

The specification and instantiation of large-scale distributed systems exploiting services presents a problem that goes beyond service composition, even more beyond the composition of service operations. One way to create such an application would be to start from a set of known services that are composed and extended by local components.

The other way is to start from a specification of the composed specification, which can be taken as an AS^2, in which most service operations are yet unknown. We may only know a categorical description for them. That is, besides search for services we also need a notion of matching services. The matching problem becomes particularly interesting, when we consider that services sought may be overlapping. For instance, when combining several booking services, each of them may contain a service operation for payment as well as one for gathering personal data. It would be not a very interesting composed applications, if such overlaps were not integrated – however, this is done in almost all approaches to srvice compositon and orchestration, in particular when BPEL is used or this purpose. First attempts to develop a theory of matching services were done in [6].

4. Conclusion. In this short position paper we outlined our view on which research challenges should be addressed in cloud computing. Naturally, these research directions describe the problems we want to address ourselves.

We identified the need to deal more carefully with concerns of cloud clients regarding loss of control, insufficient contractual guarantees, and security and privacy. Our objective is to achieve transparency in the sense that cloud-based applications should be seen by the client, as if there were no cloud involved. We envision a communicating distributed system with components located in the cloud and others on the client-side. This system will be adaptive to various cloud providers and to devices and preferences defined by the client. The key idea of our intended work plan is to specify such a system and to enlarge it step-by-step to capture access control, identification, transparent services, privacy preservation, etc.

We further plan to address cloud interoperability, which means to set up broker technology to publish and locate services in clouds, to compose them, and to run distributed applications across multiple clouds. The key idea for this part of our intended work plan is to extend the before-mentioned specification in a way that external access is enabled and controlled, and an architecture for handling such requests from outside is developed.

REFERENCES

[1] G. Alonso et al., eds., *Web Services: Concepts, Architectures and Applications*, Springer-Verlag, 2003.
[2] A. Blass and J. Gurevich, *Abstract state machines capture parallel algorithms*, ACM Transactions on Computational Logic, 4 (2003), pp. 578–651.
[3] E. Börger and R. Stärk, *Abstract State Machines*, Springer-Verlag, Berlin Heidelberg New York, 2003.
[4] J. Gurevich, *Sequential abstract state machines capture sequential algorithms*, ACM Transactions on Computational Logic, 1 (2000), pp. 77–111.

[5] H. Ma, K.-D. Schewe, B. Thalheim, and Q. Wang, *A theory of data-intensive software services*, Service Oriented Computing and Its Applications, 3 (2009), pp. 263–283.
[6] ———, *A formal model for the interoperability of service clouds*, 2011. submitted for publication.
[7] *Universal description, discovery and integration (UDDI)*. http://www.uddi.org.
[8] K.-D. Schewe, B. Thalheim, and Q. Wang, *Customising web information systems according to user preferences*, World Wide Web, 12 (2009), pp. 27–50.
[9] K.-D. Schewe and Q. Wang, *A customised ASM thesis for database transformations*, Acta Cybernetica, 19 (2010), pp. 765–805.
[10] *Simple object access protocol (SOAP)*. http://www.w3c.org/TR/soap.
[11] *Web ontology language (OWL)*. http://www.w3c.org//OWL/.
[12] *Web services description language (WSDL)*. http://www.w3c.org/TR/wsdl.

Edited by: Dana Petcu and Jose Luis Vazquez-Poletti
Received: November 1, 2011
Accepted: November 30, 2011