The Client Side of Cloud Computing

Mario Höfer, Gernot Howanitz

July 1, 2009

Seminar aus Informatik, SS 2009

Leitung: o. Univ.-Prof. Dr. Wolfgang Pree

FACHBEREICH FÜR COMPUTERWISSENSCHAFTEN, UNIVERSITÄT SALZBURG
Abstract

In this paper we propose a classification of different clients of Cloud Computing into the groups of hardware and software clients. Each one of these groups will be further subdivided in 3 classes. In a second step, we analyze two specific problems of cloud clients, namely how to discover a service, and how to decide, which service to use. Each of these problems is far from being trivial and could be essential for the future success of cloud computing.
1 Introduction

Cloud computing has become an area of very active research. With many new publications and conferences this year, the interest is very high. Because of this fact, definitions concerning cloud computing and what exactly cloud computing actually is differ. Furthermore, the client-side has not received much attention yet.

In this paper, we present a summary of our research of cloud clients with a definition of the term cloud client, a categorization of the different types, advantages and problems of cloud clients, and a little prediction what development we could see in the future.

The rest of this paper is organized as follows: We discuss one special aspect of the client side, namely the service look-up. First, in section 3.1, we present a definition of a service in the cloud context; then, in section 3.2, we present the proved technology of Web Service discovery and deliver an overview over Semantic Web problems in connection with the cloud.

Section 3.3 is dedicated to the problem, which service to use, how to estimate the Quality of Service (QoS) for one specific service, and how to automatize negotiation of business contracts with the service provider.
2 Cloud Clients

2.1 Definition

What exactly is a cloud client? We want to mention two definitions:

"A cloud client consists of computer hardware and/or computer software which relies on cloud computing for application delivery, or which is specifically designed for delivery of cloud services and which, in either case, is essentially useless without it" (Wikipedia, 2009).

A cloud client is a “interface of the cloud to the common computer user through web browsers and thin computing terminals” (Youseff et al., 2009).

So the term cloud client describes a piece of hardware, a piece of software or both, that is specifically designed for a cloud service. In the following, we will arrange the different types into groups and give examples for usage.

2.2 Hardware Clients

We will start with the different types of hardware clients. There are 3 types to distinguish between:

- **Thick Client**
  The so-called thick client consists of many interfaces, intern memory, I/O devices etc. It is a full-featured computer, which is functional, whether it is connected to a network or not. It is possible to use the thick client for many different tasks; a good example is the well-known standard desktop PC.

- **Thin Client**
  The thin client on the other hand has only the necessary components for one specific task, in the most extreme form only input and output interfaces. It doesn't have a hard drive and therefore no software can be installed on it. Instead, it runs programs and accesses data from a server. An example is the OnLive hardware.

- **Smartphones**
  Finally the third type of hardware are smartphones. They let you access cloud services from everywhere; examples are the iPhone, Android-based phones and phones with the windows mobile operating system.
Most of the cloud services available can be used with a thick client, for example the Amazon Simple Storage Service (S3), the Elastic Compute Cloud (EC2) or Microsoft LiveMesh. The Amazon Simple Storage Service is “storage for the Internet”; it provides a web-service interface to store and retrieve data in and from the cloud. This service is used for Amazon EC2, a service for resizable compute capacity in the cloud. The Amazon Machine Image (AMI) is a virtual machine that runs on EC2, it can be created by the user and uploaded to S3. The Microsoft service named LiveMesh that is currently available as Beta version, is a system to synchronize, share and access data between different computers.

In contrast the thin clients have a very specific application. OnLive is a service that is about to start end of 2009. It is ought to provide games-on-demand. The games are executed on the OnLive server that is in the cloud. The OnLive MicroConsole receives input from keyboard, gamepad or mouse and sends it to the cloud. The graphics and sound output is streamed to the MicroConsole, which displays it on a TV-set. That is all this piece of hardware is capable of.

Some cloud services can be used on smartphones, an example is the Salesforce.com Mobile Lite Client. Salesforce.com is a purely cloud based CRM system for companies.

2.3 Software Clients

Moving on to the software side, we find that software clients in general can be put in one of three groups. This different types include (in order from more desktop-related to more web-related):

- Rich or Fat Client
  Desktop applications connected to the Internet or Fat Clients are applications that make use of network support, but also run offline, some-

---

1For more information please refer to the official web site http://aws.amazon.com/s3/ [Last access: July 1, 2009]
2For more information please refer to the official web site http://aws.amazon.com/ec2/ [Last access: July 1, 2009]
3For more information please refer to the official web site https://www.mesh.com/Welcome/Default.aspx [Last access: July 1, 2009]
4For more information please refer to the official web site http://www.onlive.com/ [Last access: July 1, 2009]
5For more information please refer to the official web site http://www.salesforce.com/ [Last access: July 1, 2009]
times with limited functionality. Examples are the e-mail client Microsoft Outlook or the media player iTunes. These applications need to be installed on the user’s machine (see M.T. Hoogvliet, 2008).

- **Smart Clients**
  A Smart Client also has to be installed locally, but installation and updating is done automatically over some kind of network.

- **Web-applications/Thin Clients**
  Web-applications/Thin Clients rarely have to be installed by the user. An example is the online agenda application Google Calendar. Applications of this kind often run in a web-browser.

### 2.4 Cloud Clients

In the case of software clients for the cloud, users expect very often a browser-based interface. But if a client software has to be installed, it is most of the time of the lightweight type.

Often there are different ways to access the cloud. Live Mesh for example offers a Web based solution and a client tool. This can be different for different user-groups. For example: the end-user uses a web-based frontend to work with the cloud application, but for administration and deployment a command line tool has to be used. Sometimes it is even possible to switch to offline mode. Example for this are the Google calendar and Gmail application. After the installation of the browser plug-in Gears, they are working without Internet connection in read-only mode. New emails are sent after a new connection has been set up.

There are 3 different types of software using cloud computing.

- **web-based clients**
  The web based clients are used for example in the Salesforce.com Customer Relationship Management (CRM) system, Google Apps or Google Docs. Google Docs is an office suite, that runs in the cloud. It provides a text writer and tools to create presentations and charts.

- **client applications**
  Other systems use client applications, but often not exclusively. The

---

For more information please refer to the official web site http://gmailblog.blogspot.com/2009/03/view-google-calendar-offline.html [Last access: July 1, 2009]

For more information please refer to the official web site http://gmailblog.blogspot.com/2009/01/new-in-labs-offline-gmail.html [Last access: July 1, 2009]
Microsoft Live Mesh service for example offers a client application in addition to the web-based solution. The command-line tool available for Amazon Elastic Compute Cloud (EC2) is a set of tools written in Java that you can run from the Linux/UNIX, or Windows command line that closely mimic the Amazon EC2 API functions.

- applications with cloud-extensions
  Some desktop applications have optional extensions into the cloud, for example Mathematica and MatLab. The latest versions of the mathematics software packages MatLab and Mathematica provide an extension for compute-intensive tasks. They are capable of using Cloud Computing to perform expensive evaluations.

  To use this, one or more Amazon EC2 images have to be configured. This way even a cluster can be used to execute the calculations. MatLab offers two different workmodes batch and interactive. In batch workflow, a MatLab user can submit a job to the cluster scheduler, possibly shut down MatLab, and retrieve results later once the job has been executed. In an interactive workflow, a MATLAB user is connected directly with the MatLab workers running in the cluster. The user sends commands that are executed immediately, and the results are available as soon as the command execution is complete. Response times may be slow depending on traffic and amount of data exchanged.

2.5 Pros and cons of cloud clients

There are some obvious advantages of using a cloud client over a desktop application. One of the main advantages of cloud computing is, "that the computational work is moved from the users' terminal to data centers where the cloud applications are deployed. This in turn lessens the restrictions on the hardware requirements needed at the users' end, allows them to obtain superb performance to some of their CPU-intensive and memory-intensive workloads without necessitating huge capital investments in their local machine" (Youseff et al., 2009).

In most cases an installation of the client is not required, of course only if a web based client is available. However a plug-in-installation might be needed. No manual updates and upgrades to new version are necessary. Every computer with Internet connection can be the access point, no matter what operating

---

8For more information please refer to the official application guideline http://www.mathworks.com/programs/techkits/ec2_paper.html [Last access: July 1, 2009]
system is used. The risk of viral-infections can be reduced, if no executable is running locally.

Even if the service can be accessed with a web browser, it may be necessary to install additional plug-ins for example flash, java etc. This will not be possible for any user anywhere. Additionally, the performance and security are two essential issues. Because all data, user input, etc. have to be sent to the servers and back, delay and round trip time can be a serious problem. The same is true of security, because all passwords, credit card information, etc. have to travel over insecure networks. Finally, there is no common standard for cloud clients. Every client works only with his own service. A certain degree of interoperability would be desirable.

2.6 A look into the future

Since cloud computing is becoming increasingly important, some developments are very likely.

"Application Software of the future will likely have a piece that runs on clients and a piece that runs in the Cloud."... "The client piece needs to be useful when disconnected from the cloud, which is not the case for many Web 2.0 applications today" (Armbrust et al., 2009).

This happens already, the browser plug-in Gears makes it possible to use the Google calendar without actual connection to the Internet. If the application has a piece that runs on clients, the delay problem can be solved to a certain degree since not every input has to be computed in the cloud. The client part will likely utilize client-side rendering engines. An improved client would bring many advantages, such as:

- Better performance, because different complex processes like transactional data validation, sorting and filtering can be done without moving data between server and client.

- Immediate feedback from the user interface, because the the GUI is rendered at the client.

- Better security, since it is possible to control how much data is exchanged over the Internet.

(see Jarke and Stetter, 2005)
3 Service Look-Up

Now it is time to focus on one special aspect of Cloud Computing clients, namely how to look up services. In order to answer this question, first of all we have to define the term service for the cloud context. Afterwards we present some look-up strategies already in use as well as new ones, which are currently being developed. As some sort of wrap-around we also concentrate on additional issues which might occur after looking up a service.

3.1 Services and the Cloud

As Cloud Computing currently may be considered a hot topic in computer science, the accompanying theory is quite heterogeneous. Because of this fact, it is not only difficult to define Cloud Computing itself, but also the term service and its place and function in the cloud. Some research groups have realized this problem and try to subsume the current status quo in their reports – the quality of whose is, however, quite varying. Mei et al. (2008), for example, note that

\[
\text{[c]loud computing is an emerging computing paradigm. [...] Although the industry has started selling cloud-computing products, research challenges in various areas [...] are still unclear. Therefore, we study the methods to reason and model cloud computing as a step toward identifying fundamental research questions in this paradigm (Mei et al., 2008, p. 464).}
\]

Unfortunately, the paper fails already in defining cloud computing, because the definitions used are slightly contradictory. According to them,

\[
\text{[c]loud computing is a paradigm that focuses on sharing data and computations over a scalable network of nodes. [...] Examples of such nodes include end user computers, data centers, and Web Services [sic!] (see Mei et al., 2008, p. 464).}
\]

According to this definition, Web Services are regarded as possible nodes in the cloud. Confusingly, Mei et al. (2008) then introduce the term Service Computing, which “is an emerging paradigm to model, create, operate, and manage business services” (see Mei et al., 2008, p. 465). After presenting this definition, Mei et al. (2008) refer to Web Services only in connection with Service Computing – a paradigm which they claim differs a lot from cloud computing. Moreover, the term service is never properly defined. All these
shortcomings add up to a rather confusing, not clarifying, reading experience and emphasize the importance of sound definitions for further work.

Another approach towards this theoretical 'jungle' is presented in Vaquero et al. (2009), where all different sorts of definitions are gathered and compared against each other. In other words: This paper presents an excellent overview over the current state of research. But Vaquero et al. (2009) try to not only collect various definitions, but they strive also to extract "a consensus definition as well as a minimum definition containing the essential characteristics" (see Vaquero et al., 2009, p. 50) from them. Services may be considered being among these essential characteristics: "Many activities use software services as their business basis. The Service Providers (SPs) make services accessible [...] through Internet-based interfaces" (Vaquero et al., 2009, p. 58). According to them, the three main classes of such services are as follows:

- Infrastructure as a Service (IaaS) – Service providers offer infrastructure, e.g. for storage or processing data. This is achieved by virtualizing hardware.

- Platform as a Service (PaaS) – Quite similar to IaaS, but on a higher level. Service providers offer infrastructure plus a software platform running on it. For the customer, there is no possibility – but also no need – to interact with the virtualized hardware directly. A good example for PaaS is Google's App Engine\(^9\).

- Software as a Service (SaaS) – The highest level of abstraction. Service providers offer software services, i.e. programs, which are executed remotely. Web services are, for example, part of SaaS.

In our opinion, this broader definition of services is applicable to most cloud computing use cases. There does not seem, however, to be too great a difference between Infrastructure and Platform as a Service, so this distinction may seem to be over the top – after all, a software platform can be regarded as infrastructure. Besides this little flaw, there is another, more serious drawback to be found in this paper. Although services per se are defined and categorized, there is no connection made between them and the cloud. All we get to know is that services are in some way or other part of the cloud. Fortunately, another well-researched paper, namely Arnbust et al. (2009), helps us to put the pieces together:

\(^9\)For more information please refer to the official web site http://apps.google.com [Last access: July 1, 2009]
Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS), so we use that term. The datacenter hardware and software is what we will call a Cloud (Armbrust et al., 2009, p. 4).

Basically, this definition resembles those presented above, but there is one remarkable difference: cloud computing does not exactly equal SaaS, the former facilitates the latter. In order to make this differentiation clearer, Armbrust et al. (2009) introduce the concept of Utility Computing: “When a Cloud is made available in a pay-as-you-go manner to the public, we call it a Public Cloud; the service being sold is Utility Computing” (Armbrust et al., 2009, p. 4). Thus, an additional layer is added to the two-tier system of service consumer and service provider: the cloud provider. The purpose of this additional layer is clear: “Analogously to how SaaS allows the user to offload some problems to the SaaS provider, the SaaS provider can now offload some of his problems to the Cloud Computing provider” (Armbrust et al., 2009, p. 4). This three-layered concept of Cloud Computing is illustrated in Fig 1.

![Image](image_url)

Figure 1: The relation between Software as a Service and cloud computing. Image courtesy of Armbrust et al., 2009.

For our investigation of service look-ups in the cloud, it is not sufficient to concentrate on SaaS-related problems only. We also have to take into consideration what Armbrust et al. (2009) called the ’real’ Cloud Computing – i.e. Utility Computing. Although the IaaS/PaaS part of the investigation of
Vaquero et al. (2009) in a way resembles this concept, the authors could not adequately describe the relationship between SaaS/IaaS/PaaS and Cloud Computing.

### 3.2 Look-Up Strategies

How can one discover services of the cloud? It seems to be an easy question, but the answer is very hard to find. One – trivial – solution is to let the user manually look for services, but an automated system would be way nicer. In order to achieve this, there are some serious obstacles to be overcome, most of which are presented in this section.

As mentioned above, there are two principal situations, during which a service look up might occur. A SaaS user might want to discover services of SaaS providers, and SaaS providers might want to find those of cloud providers. At a first glance, the problem seems to remain the same in both situations. Upon more thorough inspection, however, it soon becomes clear that they differ quite a lot, especially in terms of frequency of use. Utility Computing look ups are likely to happen less often than SaaS look ups, because there will be far more SaaS providers – and even more SaaS users! – than cloud providers. The reason is simple:

While the attraction to Cloud Computing users (SaaS providers) is clear, who would become a Cloud Computing provider, and why? To begin with, realizing the economies of scale afforded by statistical multiplexing and bulk purchasing requires the construction of extremely large datacenters. Building, provisioning, and launching such a facility is a hundred-million-dollar undertaking (Armbrust et al., 2009, p. 5).

But the smaller number of cloud providers is not the only reason for a smaller frequency of use. The decision, which provider of cloud storage to use, is, for example, far more tricky than the decision, which currency converter service should be used. The storage decision problem, however, does not have to be resolved in microseconds and is likely to occur less frequently. Because of these characteristics we assume that Utility Computing look-ups do not necessarily have to be fully automated. For SaaS, however, an automatic solution is highly desirable.

Before we concentrate on current research issues in the field of (automated) service look-ups, we want to present a well-established SaaS look-up strategy with all its assets and drawbacks – the Service-Oriented Architecture (SOA) - based web services discovery mechanism UDDI (Universal Describe-
Web services are [...] SOA based technology. They use the Internet as the communication medium and open Internet-based standards, including Simple Access Protocol (SOAP) for transmitting data, the Web Services Description Language (WSDL) for defining services, and the Business Process Execution for Web Services (BPEL4WS) for orchestrating services (Ismaili and Sisediev, 2008, p. 1470).

A typical web service look-up as depicted in Fig. 2 works as follows: In the beginning, information about the web service, i.e. a WSDL document describing the service, has to be published in a central database, or UDDI registry (1). Then a (potential) customer can search this registry for suitable services (2) and download the WSDL file (3). With the information in the WSDL file, the customer can then contact the web service directly (4) and receives a response (5). All communication is done via a special protocol (SOAP), which basically exchanges XML messages via TCP/IP.

One might criticize several elements of this system. First of all, the resilience of one single centralized service registry is questionable. Actually,
one example of a centralized registry, the *UDDI Business Registry* (UBR) propagated and set up by IBM, Microsoft, and SAP, has been shut down in 2005.\(^\text{10}\) In order to create a more stable and also more scalable service registry, Makris et al. (2005) propose to use a *Peer to Peer* (P2P) network for service discovery.

Furthermore, one might criticize that this approach towards a service look-up is everything but automated. The registry has to be maintained by humans; service providers have to manually publish their services, and customers have to rely on their (personal) experience in querying large databases to find the desired service. Moreover, the selection of a service is typically conducted during compile time. On the other hand, service look-up by automated agents may also happen at run time, which allows for far superior flexibility (see Garofalakis et al., 2004, p. 5). Although automated agents can easily search for a string, they are not able to understand its semantic information. Thus, it becomes perfectly clear that automated service discovery can be considered a *semantic web* problem, which could be solved by means of using ontologies. This term is defined as follows:

> Ontologies are used to define the basic terms and relations between domains. This is the mean for data description or meta-data creation. [...] The main benefit of ontologies is in overcoming problems with semantic heterogeneities between data sources (Ismaili and Sisiediev, 2008, p. 1475).

As soon as these “semantic heterogeneities” are resolved, fully automated web service look-up is no longer a problem. Currently, there are several approaches towards this problem. Crasso et al. (2008), for example, suggest to query web services by example (*Web Service Queries by Example* (WSQBE)). They also propose a new method to reduce search space. Dietze et al. (2008), on the other hand, want to extend the concept of *Semantic Web Services* (SWS), which “enable the automatic discovery of distributed Web services based on comprehensive semantic representations” (Dietze et al., 2008, p. 1). According to them, SWS technology works great as long as the task, which is to be performed, is well-defined. As soon as a client searches for services for a specific *situational* context, SWS reaches its limits. In order to overcome these limits, Dietze et al. (2008) introduce *Conceptual Situation Spaces* (CSS), which are defined as follows:

> CSS enable the description of situation characteristics as members in geometrical vector spaces following the idea of Conceptual

---

\(^{10}\)Microsoft explains this measure in the *UBR Shutdown FAQ*, which may be accessed online: [http://uddi.microsoft.com/about/FAQshutdown.htm](http://uddi.microsoft.com/about/FAQshutdown.htm) [last access: July 1, 2009]
Spaces. Semantic similarity between situations is calculated in terms of their Euclidean distance within a CSS. Extending merely symbolic SWS descriptions with context information on a conceptual level through CSS enables similarity-based matchmaking between real-world situation characteristics and predefined resource representations as part of SWS descriptions (Dietze et al., 2008, p. 1).

We hope that this section was able to provide a profound overview over current cloud problems and solutions. Evidently, the connection of Cloud Computing and Semantic Web technologies are both one of the most promising and most demanding tasks for pushing cloud development forwards.

3.3 After Look-Up

We have just proven that looking up cloud services might be problematic, but even after finding a service there are still issues to be overcome. One important question is, which service to use if multiple services are returned by the discovery mechanism. The services need to be ranked by quality metrics. Armbrust et al. (2009) have identified the top 10 obstacles for cloud computing, among them Availability of Service, Data Transfer Bottlenecks, Performance Unpredictability, and Bugs in Large-Scale Distributed Systems. These obstacles have an direct impact on the Quality of Service (QoS), so this measure is one of the most important factors in determining which service to use. In our understanding, QoS is one of the key attributes of a service. These numbers, however, are not retrieved automatically; they have to be calculated. Diamadopoulou et al. (2008a) propose a framework, which adds QoS measures to standard Web Services. This could be easily adapted to fit any other cloud service remotely.

Another factor is – of course – money. Service providers eventually might want to make money with their services. There is, of course, the possibility of offering batch contracts – e.g. 10 uses of any service the provider has in his catalog – or time-based contracts – e.g. a ‘flatrate’. The most flexible solution, however, would be to offer pay per use contracts. This implies, however, that at least once, upon service discovery, some sort of agreement has to be reached – and it would be best if this was conducted automatically. Brandic et al. (2008) propose a system which facilitates automatic negotiations of Service Level Agreements (SLA) between service providers and consumers. They describe a structure very much like the UDDI approach. One point becomes absolutely clear: The problems in the cloud seem to be always the
same ones; there is no chance to automate the process as long as semantic extensions are not part of the *cloud*. 
4 Resumé

*Cloud Computing* is an emerging topic of computer science, and thus, theory building at best can be regarded as 'being in progress'. A number of *cloud* topics is either not addressed by today's research groups – as far as we know, the paper at hand, for example, is the first try to categorize cloud clients – or a vivid discussion as regards their definition is still an ongoing one. This latter problem has been illustrated in the second half of our paper, where we tried to produce a composite definition of *services* in the cloud context. Then, we proceeded in pointing out the main issues during service discovery, a research topic, which is far from being trivial. In our opinion, further automatization of service discovery and negotiation of service contracts is essential for a throughout success of the *cloud*. As long as the *Semantic Web* is only a theoretical concept, *Cloud Computing* will not be able to reach its full potential.
References


19


