Lecture Summary 5: Cloud Computing

This summary was intended to be divided into two parts, first a discussion of cloud-computing in general with its own evaluation. And a technical considerations for a specific mobile-oriented implementation of Amazon's cloud service, I decided to take the Amazon part out and delegate it as-is to the previous write-up.

Cloud Computing then, or *the Cloud*, are originally marketing terms referring to service-oriented computing, pertaining to the translation of computing resources from localised devices to distributed services delivered over the Internet.

The terms are vague because of their generality, a cloud service can relate to any type of computational activity that is accessed over a network. It is also an evolutionary concept without any absolute objective. A useful way to think about it is the increasing trends of applications, that are moving away from being processes on our desktops into web-services that can be independently accessed by any number of clients. One of the first classes of applications that experienced this shift were desktop based email clients; today the majority of people use fully web-based clients. This evolution pre-dates the term "cloud", so it may both be applied in retrospect, and with future developments in mind as a longer term evolution in the history of computing.

In addition to the shift of many applications that are personally-orientated, such as scheduling tools, contact management and photoalbums. Applications that are dependent on multiple-users are only meaningful if they are cloud-enabled, such as social-networking and collaborative word-processing.

In terms of this discussion we need to consider if, or what, computational tasks are more suitably delegated to localised devices, and inquire if computational resources could possibly be entirely converted as a pervasive utility, analogous to how electricity once established itself as a commodity resource. In the context of mobiles we should consider if the cloud is more, or less relevant, and in what ways their suitability as clients differ from the desktop. Let's consider some properties of the cloud in this context, one aspect that may be considered a drawback of all the following is the need for network availability and bandwidth (variable), I'll leave mention of network connectivity until the discussion.

Offloading Storage Relieving clients of the need to store their own data is good for persistence, and saves space. Bandwidth requirements may be large depending on the application, for example a remote file-server. Google Email and Contacts provide ample online storage. In case one loses ones phone the data can be downloaded again, and only data that one is immediately interested in needs to be downloaded. Many large-scale applications could only work by providing the client by small amount of relevant data at the time. In Google's satellite images the data requirements are enormous. Google stores the world's land-based areas at an average of about 300kb per km^2, which translates to about 450TB (there is 150 million km^2 of land on Earth); more data than any single device is able to contain, Clients only downloads a tiny section of the world as needed, the data is in addition continuously updated by numerous sources (an example of *orchestration*).

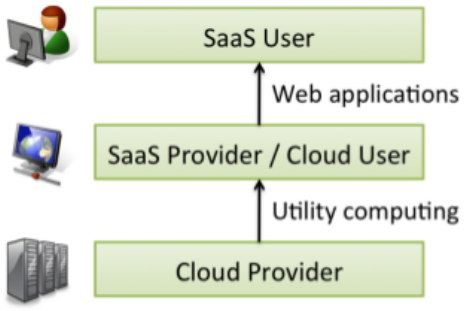

Figure 1: Amazon EC2 is a cloud provider, SaaS providers build their applications on the resources (and optional libraries) that Amazon makes available.

Offloading Processing Many applications are compute intense, examples are computer vision, voice recognition and natural language translation. In these cases a client may upload the data to a server for processing, and wait for a computed response. Considerations are the ratio of bandwidth required compared to the saving on processing, and also real-time requirements. If we want the phone to perform feature recognition (computer vision) in real-time for example, then both an unacceptable latency may occur, and the relative costs between bandwidth compared to savings on processing may be undesirable. On the other hand, a phone has limited computational resources which make remote processing more attracting compared to the desktop. And, some processing tasks may additionally be dependent on data that is only available on the server, such as statistical databases used to aid natural language processing.

Orchestration Some complex systems are shared between many clients. For example the search engine Yacy[1] which distributes workloads such as search crawling, indexing and algorithms among an unbound number of participants. Another example is the decentralised digital currency Bitcoins[2] which relies on different clients verifying the integrity of each other. These have

[1] http://yacy.net/en/
[2] http://bitcoin.org/

the properties of automated coordination and management, and (in the examples mentioned) with limited requirements for centralised control. Generally, orchestration involves the collaboration of many clients, that are peers with respect to the given application. A simple example of orchestration are online forums on some given topic, this enables discovery of clients that have registered their intent to be found.

discussion

An inherent aspect of cloud computing is the necessity of having a network connection for it to be a meaningful concept, so, it's not accurate to call this a drawback because the cloud per definition assumes network connectivity. It is better to talk about bandwidth, and in a pragmatic sense we can accept that bandwidth is at periods unavailable. The services that we delegate to the cloud, and its increasing ubiquity, are growing in-step with growing pervasiveness of network availability.

I believe this is the case because the cloud, in general (given availability and bandwidth), has universally beneficial properties, many of them related to management of data. In the cloud-model, data is stored centrally which gives it the benefits of pervasiveness, persistency and ability to share. Possible drawbacks may be related to privacy and security, but these are largely solved as demonstrated by all the current web-services that we already depend on.

An implication of the model is that clients simply become access points for viewing, collecting and modifying data. They are thin-clients, and they become replaceable. In respect to mobiles these are properties that are particularly advantageous. For example. increasing the number of personal computing devices could potentially lead to fragmentation of our personal data. But it is noticeable that this is almost entirely avoided in all the services that we are using mobiles for today, and we can see that almost all applications on mobiles today are in fact fully cloud-enabled.

On the question raised if there are any computing tasks that should still remain in the domain of the client, the answer is yes, but many of these are based on pragmatic considerations while network ubiquity is still in its developing stages, such as a necessity for caching data in case the connection fails. In a longer term perspective where connections will become faster and more reliable, client-side processing will remain in many forms related to reasons of efficiency.

For the application developer, many of the practical challenges of the cloud are the same as the one considered with respect to network connectivity (lecture 2). The more general class of challenges are related to the aspects above on offloading storage, processing and orchestration, to identify further areas of innovation, and to develop with the correct delegation of responsibility between the client and the server.

References

[1] Armbrust M. and et al. Above the clouds: A berkley view of cloud computing. *UC Berkley Reliable Adaptive Distributed Systems Laboratory White Paper*, 2009.

[2] Buyya R. and et al. *Cloud Computing, Principles and Paradigms*. Wiley, 2010.