INTRODUCTION

In this chapter we introduce the context of Big Data and some examples of Data Overflow problems related to it. We then highlight some key aspects of Cloud Computing, mostly related to Cloud Storage, that will be useful to this work. Finally, we use the given example scenarios as a motivation to propose an abstraction that uses the Cloud environment to deal with Data Overflow in electronic devices.

1.1.Context

1.1.1.Information Proliferation

The advent of Social Networks brought together millions of users, and many of these users have integrated the use of Web applications for social networking into their daily practices [1]. Current advances in mobile computing technology allowed for the integration of such applications with a plethora of hardware devices, and it is fair to say that widespread use of Social Networks directly influenced the adoption of mobile devices. It is common to spot people all over the World using smartphones, tablets and notebooks to take pictures, record meetings and make videos so as to share their social experiences.

Advances in Scientific Research on the last decades consolidated the practice of *in silicio* experiments in a science paradigm strongly oriented towards data exploration: data is captured by instruments or generated by simulators; then it is processed by software and stored in large databases for future analysis using data management and statistics [2].

On the Business Intelligence arena, old and modern techniques of aggregating, analyzing and presenting various different types of data in succinct yet expressive ways are regarded as essential core competencies for strong competitive market players [3].

What these three phenomena, at first glance so disparate in nature, have in common is that they all account for an impressive growth of digital data. As a

consequence, they put forth a demand for scalable new ways of processing, storing, retrieving and analyzing massive amounts of data.

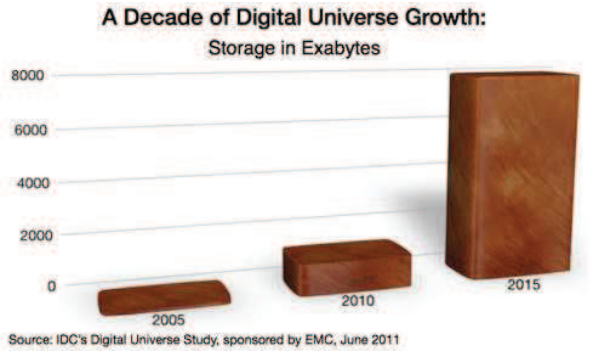

Figure 1: Storage growth expected to more than double a year

1.1.2.Software as a Service and Cloud Computing Origins

Parallel to the increasing demand for data storage, the idea of Software as a Service has also been steadily maturing over the last decades, and important players in the software industry, e.g., IBM, HP, Microsoft, have been shifting gears towards this business model.

One possible definition for Software as a Service, made popular by SalesForce.com, is the following:

Software as a service (or SaaS) is a way of delivering applications over the Internet—as a service. Instead of installing and maintaining software, you simply access it via the Internet, freeing yourself from complex software and hardware management. SaaS applications are sometimes called Web-based software, on-demand software, or hosted software. Whatever the name, SaaS applications run on a SaaS provider's servers. The provider manages access to the application, including security, availability, and performance [4].

It is in this context that Cloud Computing blooms; from initially proposed "Software as a Service" architectures which were meant to be comprehensive and yet proved cumbersome to use by end users [5], to nicely packed, ready to consume, billable services that leverage advanced data storing, networking, and

virtualization techniques that shield infrastructure management concerns from end users. Users are able to, henceforth, run their software and store their data in remote datacenters through simplified Web interfaces or APIs defined by the Cloud providers of their choice.

1.1.3.Cloud Computing in Software Processes

The National Institute of Standards and Technology defines Cloud Computing [6] as follows:

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

A comprehensible definition for Cloud Computing is a subject of much debate [7]; in what follows we provide working definitions for concepts that are central to the understanding of this proposed work:

• Elasticity – Resources can be provisioned and quickly disposed to accommodate the needs of different business models. In some, the need for resources can be fairly stable over time, whereas in others usage peaks demand more resources. In any case, however, end users are under the impression that they have access to unlimited resources. That is possible because providers maintain resource pools that serve multiple consumers, with different physical and virtual resources that are dynamically assigned based on consumer demand.

• Internet access – resources are accessible over the Internet via well-established communications protocols; e.g. URLs and standardized Web Services endpoints.

• Billable, pay as you use, on demand self-service – Users can buy Cloud services, such as server virtualization, storage or compute power according to general Service License Agreements without the need of individual contracts tailored for each user and provider. Moreover, a pay-as-you-use model allows for

recurrent billing (e.g. monthly, yearly), which usually include access to resource usage management tools and metrics.

Different Cloud provider implementations will imply different levels of service, but the above key concepts are dear to most Cloud service providers – otherwise we shall argue that it is NOT the Cloud, and therefore are outside the scope of this work. Also, in most Cloud implementation, having users' assets (data, software etc.) running on Cloud-ready datacenters will secure their investments with automatic backups. Additional benefits include defective hardware replacement, fault tolerance, load balancing, automatic OS and patch updates.

If we consider, at one side, all features offered by Cloud systems, and at another side, the aggressive increase for on-demand for computing resources combined with the innovative ways to produce and access content using mobile devices, it will be easy to understand why the Cloud is rapidly becoming the principal platform in which to store and retrieve digital information; the Cloud offers both secure storage and processing power for most scenarios dealing with massive digital data. As a consequence, we'll soon be living on a world where Cloud Computing will be crucial to cater for Society's needs related to information storing and utilization, and Cloud Computing will play a central role in most Software Engineering practices.

In response to all this hype and excitement about Cloud Computing, several lines of research lines geared towards integration of the Cloud into software practices are starting to appear. As a contribution for the realization of this vision, i.e., that of Cloud-driven Software Engineering processes, this M.Sc. thesis aims at providing a conceptual model abstraction that integrates **Cloud Storage** features into the design phase of a software development life cycle. In particular, we are looking into defining useful constructs that allow the incorporation of Cloud capabilities in both traditional and emerging conceptual model designs of software applications.

1.2.Problem Scenarios

An ever increasing number of Internet applications deal with collecting, storing and retrieving media files, e.g., music, pictures and scientific measurements.

Consider the following scenarios:

1. A Mobile Phone seller wishing to boost its sales is launching the **OPS – Overflow Protection System**. Mobile Phone hard disks aren't particularly famous for offering huge storage capacity. To ensure to the customers that they will never run out of storage space while doing important tasks, the seller will offer a limited time, renewable guarantee that even if they run out of local space to store new pictures, songs, videos or other files, these overflowing files will be seamlessly directed to an external storage, and they will still be able to see and manipulate those files as if they were stored locally, using their proprietary software to browse their files as well as the files content. The same proprietary software will be able to reclaim the files from the external storage when local storage is again available.
2. The Austrian government is preparing a huge event for 2012 when Joseph Haydn's 280 birthdate will be celebrated. They will create a Web Site where String Quartet groups can upload a video to a Web server local folder where they perform a piece composed by Haydn, along with information about each player. The general public can use the site to search uploaded videos by musician birth country, musician name or piece title, then download and watch the selected videos. The best submissions will win a prize.

In neither scenario it is possible to estimate the required storage space for running the software applications. Consequently, there's a risk that the client applications simply run out of storage. In [8] we discuss the implications of such problems, baptized as "open ended", for the storage and computational power provisioning challenges they impose. In scenario 2 one could plan for the worst-case scenario, by buying upfront a great amount of storage, which could turn into

a great waste of money, both for buying and maintaining the equipment. Even so, the estimated amount of storage could be still insufficient. In scenario 1, hardware limitations on Mobile devices make even this strategy impossible.

We need therefore to come up with a solution that deals gracefully with **data overflow** without incurring in buying storage/resources upfront to end up with unused resources, and Cloud computing infrastructures are viable solutions for open ended scenarios such as the ones described previously [9].

1.3.Work Contribution

The main contribution of this work is the proposal of a database management architecture that uses the Cloud environment on a simple programming model to deal with unpredictable demand or unexpected shortage of local computer storage, by exploring the concept of *Elasticity* provided by Cloud Computing environments.

We present this work in three layers – Conceptual, Design and Implementation – and each layer is a software artifact on itself.

In Chapter 2 we list some key underlying technologies that were used to realize proof-of-concept implementations of our proposed solution, with brief justifications of their choices. This chapter can be overall skipped if the reader is familiarized with current Microsoft APIs/technologies for software development.

In Chapter 3 we present the Conceptual layer, the **Container Database (CDB)**. This abstraction is free of implementation details and could be integrated in any software product that handles local storage but needs to address occasional data bursts.

In Chapter 4 we present the Design layer. We propose an Object-oriented design as a base blueprint for the Container Database abstraction, describing its main components, operations and mechanisms.

In Chapter 5 and 6 we present the Implementation layer for Desktop PCs and mobile devices, respectively. We describe an API for the Container Database to be used in applications that can be directly used in Client Applications.

In Chapter 7 we present some conclusions about this work and we point to some directions for future work.

In Chapter 8 we present a brief discussion on related works.