Chapter 9
Migration of Data between Cloud and Non–Cloud Datastores

Shreyansh Bhatt
DA-IICT, India

Sanjay Chaudhary
DA-IICT, India

Minal Bhise
DA-IICT, India

ABSTRACT
The on demand services and scalability features of cloud computing have attracted many customers to move their applications into the cloud. Therefore, application, data access, storage, and migration to and from cloud have garnered much recent attention, especially with well-established legacy applications. Cloud service providers are following different standards to host applications and data. In the present chapter, the authors focus on data migration from various datastores to cloud and vice versa. They have discussed various challenges associated with this reciprocal migration and proposed a simple yet powerful model whereby data can be migrated between various datastores, especially cloud datastores. The results show an efficient way to move data from conventional relational databases to Google App Engines and how data residing in the Google App Engines can be stored on relational databases and vice versa. They provide a generalized architecture to store data in any cloud datastore. The authors use RDF/RDFS as an intermediate model in the migration process.

INTRODUCTION
This chapter introduces the data interoperability problem associated with cloud computing. The chapter begins with introduction to various cloud datastores to emphasize how their different data storage schemes affect data interoperability. RDF/RDFS has been used to provide interoperability in Web services (Auer, 2010) and in relational database merging (Nagarajan, 2006); we have used it to provide interoperability in cloud data migration. Mapping algorithms are discussed in detail to facilitate the migration. We have used

DOI: 10.4018/978-1-4666-2488-7.ch009
Migration of Data between Cloud and Non-Cloud Datastores

mySQL as non-cloud datastore and the Google App Engine as cloud datastore to show the migration process. We have further proposed the generalized framework for cloud data migration. The main objective of this chapter is to provide a solution to migrate data between datastores with different schema. In other words, we have aimed and achieved data and schema migration.

MAIN FOCUS OF THE CHAPTER

Cloud Datastore

Various cloud datastores and their data storage schemes are discussed. An application running on the cloud requires dealing with a huge amount of data. Applications must be as scalable as the database of application. RDBMS is not suitable for such a requirement, e.g. handling a large amount of unstructured data, providing elastic scalability, etc. Therefore, new document oriented distributed datastores are emerging to cater to these requirements. Moreover, different cloud datastores are following different schemes to store the data. We discuss the data storage scheme of such cloud datastores to elaborate the point.

Apache CouchDB

CouchDB is an open source document-oriented schema free database-management system, accessible using a RESTful JavaScript Object Notation (JSON) API. Couch stands for Cluster of Unreliable Commodity Hardware. It stores data in JSON format. It can include all native datatypes in programming language. It stores data in the form of a document.

CouchDB is schema free, i.e. one document can have fields that another document doesn’t have. Documents are the actual representation of data objects. CouchDB stores uniquely named documents with document ID and revision number. Each document can be made of a number of fields not bound by size and with unique names. Documents can have attachments that can be both text as well as digital. When changes are made, a new version of document called revision is created. It does not have a locking mechanism. Two users can load and edit the same document at same time. It maintains consistency by ensuring updates either work or fail (Bhat, 2010).

It achieves scalability and availability by periodically copying documents between servers. It supports file attachments, which can be in the form of music, images, etc. This feature is not seen in traditional databases. It assigns a universally unique identifier to each and every document. It does not support join. It allows creating arbitrary relations between documents.

Amazon SimpleDB

Items are stored as resources in simpleDB. These resources can have multivalued attributes. These resources can be related to each other, and the relationship between these resources can be visualized as a hierarchical tree structure as shown in Figure 1.

As it is a schema-less storage, an item can have a different set of attributes from other items in the domain. It does not store raw data. Rather, it expands input data and creates indices over multiple dimensions to quickly query that data. It can be used as a flat file store. Individual item name, attribute names, and attribute values can be up to 1024 bytes in length. Amazon simpleDB allows 10GB of storage for each domain with 100 domains per customer account, which provides 1TB of total storage.

Bigtable: Google App Engine Datastore

The most popular way of persisting data in Web applications has been the use of relational databases.
Related Content

LDAP Vulnerability Detection in Web Applications
[www.igi-global.com/article/ldap-vulnerability-detection-in-web-applications/204523?camid=4v1a](www.igi-global.com/article/ldap-vulnerability-detection-in-web-applications/204523?camid=4v1a)

Testing in the Cloud: Balancing the Value and Risks of Cloud Computing
[www.igi-global.com/chapter/testing-cloud-balancing-value-risks/72242?camid=4v1a](www.igi-global.com/chapter/testing-cloud-balancing-value-risks/72242?camid=4v1a)

Advancing Cyber Resilience Analysis with Performance-Based Metrics from Infrastructure Assessments
[www.igi-global.com/article/advancing-cyber-resilience-analysis-performance/76356?camid=4v1a](www.igi-global.com/article/advancing-cyber-resilience-analysis-performance/76356?camid=4v1a)

Backstepping Control for Synchronizing Fractional-Order Chaotic Systems
[www.igi-global.com/chapter/backstepping-control-for-synchronizing-fractional-order-chaotic-systems/204799?camid=4v1a](www.igi-global.com/chapter/backstepping-control-for-synchronizing-fractional-order-chaotic-systems/204799?camid=4v1a)